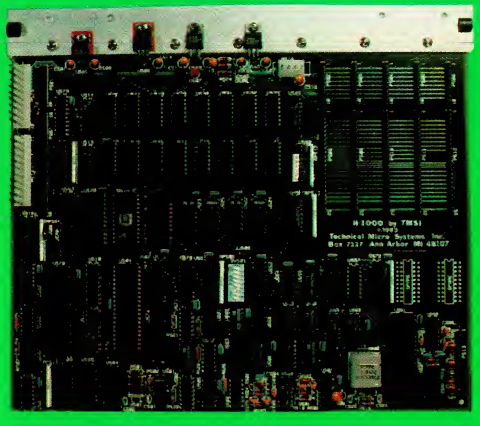
SEXTANT®

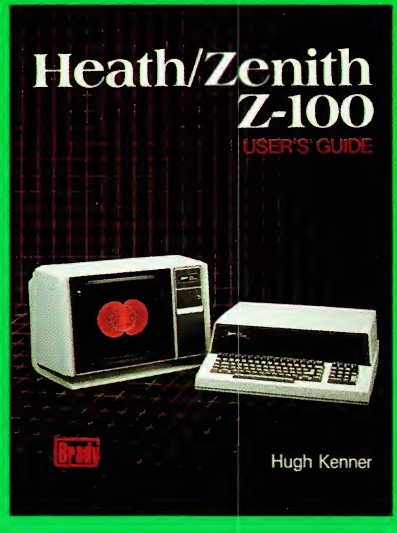
$2.95
January-February 1985
Issue No. 14

The Independent Magazine for Users of Heath/Zenith Microcomputers



TMSI's H-1000: Grafting an 8086 Onto Your '89

Hugh Kenner's Book for the Z100 Beginner



Microsoft Word and Mouse: Do They Click?

WatchWord: Custom Built for You and Your Z100

W.I.S.E.'s I/O and Clock Board for the '89

SEXTANT

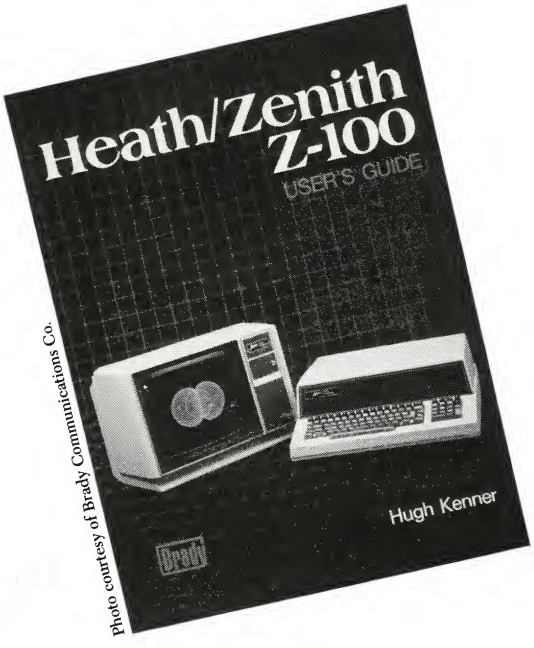


Photo courtesy of Brady Communications Co.

31 It's finally arrived, two years behind the computer. Many of us wish we'd had it when we were starting with our Z100s. But it *is* out in time for those of us who are still learning. It's a comprehensive book for the beginner, covering all aspects of using a Z100. (It's taught our reviewer everything he knows about ours.)

Features

69 Jazz up an uninteresting program. Entertain your guests. Frighten burglars (if they're willing to read your screen display). The applications for this electronic billboard are endless. As your message crosses the screen, who will believe it was achieved in only two lines of programming in MBASIC?

Issue #14
January-February 1985

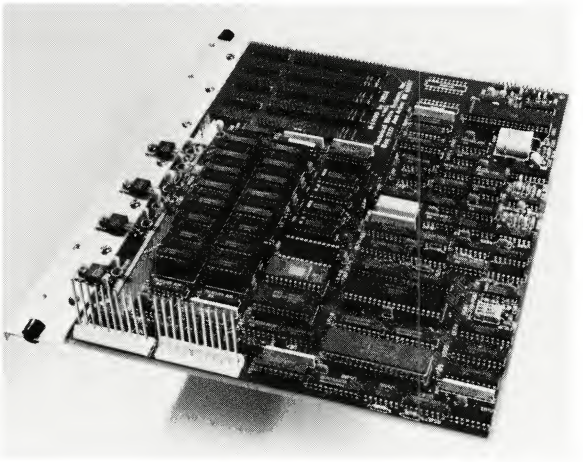


Photo courtesy of TMSI

17 Many H/Z89 owners want to be able to run 16-bit software, but are reluctant to trade in their computers. Some of them have bought Z100s, knowing that if they don't have much use for the older machine, at least they can use a lot of its software. The H-1000 has many of the same qualifications as the '100—inside the '89 box. Is this the answer?

Departments

Cover Photos:
H-1000 board courtesy of TMSI
Z100 book courtesy of Brady Communications Co.
Mouse courtesy of Microsoft Corporation

the H-89 TWOET SYSTEMS

The H89 TWOET systems Just Got Hard to beat!

We've got a great idea for your H-88, 89 or 90. It's a dual internal half height drive system. Two of our half height 5¼" drives can replace your built-in disk drive, doubling your information storage capacity.

Floppy Disk Services provides you with everything you need. That's two double-sided, double density, half height drives in either 48 or 96 tpi format, all hardware, cables and power connector adaptors. And almost important, you get easy, step-by-step instructions, in the Heath/Zenith tradition of good, clear documentation.

We've thoroughly tested the TWOET/Heath set-up. And now by popular demand, the new 10 megabyte hard disk system! We are now able to offer you the storage you want and need for a fair price! The hard disk system comes with all software needed to run with the 17 or 37 controller and/or the Magnolia Double Density controller.

NEW!

Model TWOET 455
2 Shugart SA-455 half height
48 tpi double sided
All hardware
Metal, shielded mounting plates
Data cable with chassis connector
Power "Y" connector
Complete documentation
Price.....$483.00

Model TWOET 465
2 Shugart SA-465 half height
96 tpi double sided
All hardware
Metal, shielded mounting plates
Data cable with chassis connector
Power"Y" connector
Complete documentation
Price.....$585.00

HARD DISK
For
H–89
Z–150
CALL!

SA-860 DS/DD half hgt 8"	505.00 2@ 495 ea.
SA-455 DS 48tpi ½ hgt 5.25"	245.00 2@ 230 ea.
SA-465 DS 96tpi ½ hgt. 5.25"	295.00 2@ 275 ea.
FDD-100-5C3 floppy 5.25"	***Sale 99.00***
FDD-100-8 SS/DD 8"	125.00
Teac 55B 48tpi DS/DD half height	195.00
Teac 55F 96tpi DS/DD half height	250.00
FDD-100-8 SS/DD 8"	120.00
FDD-200-8 DS/DD 8"	***Sale 185.00***
Microscience 10 MB half height hard disk	725.00
PGS, HX-12 RGB color monitor	525.00
Data connectors of all types	***CALL***
Power connectors for all drives	***CALL***
Controllers	***CALL***
Hard disk Z150	Internal 895.00

Wondering what to do with your internal drive if you buy this system?

Here's the solution. If you purchase a dual half height system for your Heath computer from Floppy Disk Services, just include an extra $60.00 plus shipping and receive a single 5¼ case with power supply and data cable ready to receive your SIEMENS internal drive! the case with data cable is normally a $80.00 item. And the cable that comes with your TWOET system includes the external chassis disk I/O connector.

Due to production deadlines, prices in this ad are 2 months old, so we encourage you to call us for current prices and new product info. Prices and specs subject to change without notice.

Dealer inquiries invited.

PAYMENT POLICY – We accept MasterCard, VISA, personal checks and Money Orders. We reserve the right to wait 10 working days for personal checks to clear your bank before we ship. All shipping standard UPS rates plus shipping & handling. NJ residents must add 6% tax.

Toll Free Order Line:
(800) 223-0306

Tech Help or Info:
(609) 799-4440

FLOPPY DISK SERVICES™ INC.

39 Everett Drive Bldg. D, Lawrenceville, NJ 08648

H-88/89/90 and Z-150 are registered trademarks of Heath Corp.

Circle #117 on Reader Service Card

SEXTANT

Charles Floto
Editor
Publisher

Sharon E. Conaway
Assistant Publisher

Jennie D. Allen
Managing Editor

John Walker
Technical Editor

Mark Goodkin
Staff Writer

John Gill
Subscription/Retail
Service Manager
Production Assistant

Bryan M. Rutberg
Advertising Sales
Assistant

Gabriele Burkhard
Circulation Assistant

Subscription Service • 202/544–0900
Advertising Inquiries • 202/544–2868
Authors' Inquiries • 202/544–2867
Charles Floto • 202/544–0484

Sextant (ISSN 0731–2180) is published bimonthly by Sextant Publishing Co., 716 E Street SE, Washington, DC 20003. Second-class postage paid at Washington, DC and at additional mailing offices. POSTMASTER: Send address changes to Sextant, 716 E Street SE, Washington, DC 20003.

Subscription rates: 6 issues (1 year) in U.S.—$14.97; in Canada and Mexico—$17.25; overseas—$21.00. Payment accepted by check in U.S. dollars payable on a U.S. bank, by U.S. and international postal money order, and by Visa and MasterCard. Please allow six to eight weeks for delivery.

The Editorial Eye

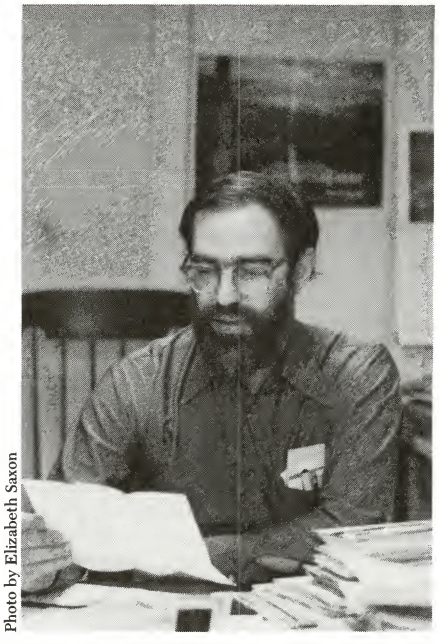


Photo by Elizabeth Saxon

We're happy to start 1985 by bringing you an issue of *Sextant* that's 16 pages bigger than each of the previous four issues. About half of the increase represents more advertising.

In September, Bryan Rutberg came on board to help us foster communication between *Sextant* readers and vendors serving the Heath/Zenith community. He's gotten off to a good start. We hope to be bringing you an increased flow of advertising throughout the new year.

The increased advertising in *Sextant* is the most visible sign of our financial health. Several computer magazines went out of business during 1984. It's probable that several others will fail to survive 1985. We want to continue serving you—and want you to know we're in good shape to do so.

Things do sometimes get a bit busy around here. The situation is compounded when events call us out of town. There were four such events between mid-October and mid-November—about as many as in the other 11 months of the past year combined.

The first was the Air Force Small Computer Conference in Montgomery, Alabama. John Walker, Mark Goodkin, and I introduced *Sextant* to hundreds of military users of Zenith Data Systems computers—primarily the Z120. We also got to talk to a good number of current subscribers.

The first Saturday in November found our entire staff at the third annual conference sponsored by the Capital Heath Users' Group. CHUGCON 84 was just across the river in northern Virginia, but it did require a bit of preparation.

The following weekend, five of us had a nice trip to the Western Regional HUG Conference. Bryan Rutberg and Mark Goodkin left a few days early to visit relatives in southern California. Jennie Allen, John Gill, and I flew out Friday and returned to the East Coast Sunday night.

The five of us and Sharon Conaway spent various of the next few days in New York at a publishing conference. We attended seminars on reporting, ad sales, photography, magazine management, design, production, and selling subscriptions.

It's good for us to get out of the office periodically to keep in touch both with the rest of the Heath/Zenith computer community and with the rest of the magazine publishing industry. It would be nice, though, if 1985's trips could be a bit more spread out.

Charles Floto

Letters

KEA Responds to ZSTEM Comments

Further to your Z-DOS Communications Roundup [*Sextant* #12, September-October 1984], I was concerned about a number of points. I appreciate that it is difficult to say much about a package if it doesn't run; however, I did invite you to call if you had any questions. I think it was unfair that you contacted other suppliers regarding their problems so they could be explained or resolved but did not contact us regarding your problems with ZSTEM.

Of the thousands of packages we have in the field, we have had only one other case of "unrecognized BIOS." This was by one of our contacts within Zenith in Chicago who was using a "non-release" version of Z-DOS. I suggest you must have a very old "preliminary version." I agree that it is unfortunate ZSTEM is release-sensitive. The sensitivity is caused by the fact that we bypass the operating system extensively to ensure maximum performance and throughput. We ensure that our products run on all official releases (including Version 2).

Also, I had included a note regarding the availability of ZSTEM on the Z150. You mentioned one other package runs on the Z150. You did not mention ZSTEM on the Z150.

Thank you for mentioning that we have at least one happy user. I think you realize how important reviews are for us in the marketplace and I'm sure you appreciate our concerns.

Eric Alexandre
KEA Systems Ltd.
No. 311–811 Beach Ave.
Vancouver, B.C., Canada V6Z 2B5

We regret that Mr. Alexandre was not contacted regarding the BIOS problem with our copy of ZSTEM. This was an editorial oversight, for which we apologize.

Regarding the availability of ZSTEM for the Z150, we received this information too late to include it in the survey.

. . . And a Reader Agrees

In your recent review of communications software [*Sextant* #12, September-October 1984] you reported that you were unable to run ZSTEM because of a BIOS problem. This may have given the impression that it is difficult to use. I have used both ZSTEM-VT100 and ZSTEM-Hobby on a Z110 and a Z120 and have been very pleased with them. As far as I can tell, they do exactly what they are advertised to do. The ability to define "softkeys" to send any sequence of characters (not a part of ZSTEM-Hobby) gives you a method of defining arbitrary dialing and login sequences. The file-transfer support (ASCII or XMODEM) is straightforward and easy to use. The VT-100 emulation is as complete as the Z100 hardware allows, omitting only 132-character lines and double-height or intensified characters. The documentation is clearly written and complete. The novel anti-piracy scheme you described is a good one, although users with light pens would have to do some plug swapping when they wanted to run ZSTEM.

Alan T. Moffet
Los Angeles, CA

Hilgraeve on ACCESS

Sextant's recent review of communications programs (September-October issue) portrayed Hilgraeve's ACCESS as a full-featured communications program for use with CompuServe and other utilities, with bulletin boards, and mainframes. I'd like to point out a singular strength that wasn't mentioned—ACCESS's powerful support of micro-to-micro communications.

Versions of ACCESS for the H8, H/Z89, H/Z100 and H/Z150 all have a menu-selectable mode that lets these computers communicate smoothly with any other micro, by cable or modem. This mode makes file transfers easy and error-free. When in this mode, a micro equipped with ACCESS need not be attended, because it responds to commands sent by the operator of the remote micro (which may use ACCESS or almost any other communications program).

A second micro-to-micro mode was added in August 1984 to the MS-DOS versions of ACCESS: an unattended, password-protected host mode that automatically matches the baud rate of the caller. ACCESS keeps a log of incoming (and outgoing) calls.

A rare and highly sought capability was added to the MS-DOS host mode in November 1984: Callers who have a special password get unrestricted use of the host system, just as though they were sitting at its keyboard. They're free to execute operating-system commands or run other host-resident software such as editors, compilers, and data-base programs.

Interestingly, the host feature isn't embedded in ACCESS, but is a script written in ACCESS's autoexecute language, AP. Source code for this and other scripts comes with ACCESS, along with full details on AP. This means a user can modify the host script to create his own custom host system or bulletin board.

ACCESS is available from almost every Heathkit Electronic Center, from many ZDS dealers, and can be purchased directly from Hilgraeve Inc. (313/243–0576).

Matthew Gray
Hilgraeve Inc.
P.O. Box 941
Monroe, MI 48161

Can't Fault Zenith

I do agree with your editorial that there seems to be a lack of direction between Heath and Zenith. I cannot, however, fault Zenith for following the path of greatest profits. The balance sheet and their shareholders certainly exercise some influence over that decision. It's only good business to make money, as you well know. Although the proliferation of Zenith-only retailers may have cut into Heathkit's sales, you have failed to acknowledge the benefits brought to the users.

First, the added Zenith outlets have brought competition into our marketplace, often resulting in lower prices to the end user. Even Heathkit has lowered its prices. I am typing this letter on a '150 system that I would not have purchased from Heathkit were the price not competitive. With the HUG discount and their current bundled software, the computer comes as a system with all you need to start at a rather attractive price.

Second, if Zenith does take over Heath's production facilities, this may push Heath into producing computers more tailored to its users and less tailored to the masses. The only true travesty that could come would be the Heathkit stores degenerating into another chain of "ComputerLands."

Thirdly, as you well know, the expansion of Zenith's market penetration has brought other suppliers into the Heath/Zenith fold. Looking at the current issue of *Sextant*, I note that the old-line suppliers to the Heath/Zenith community have not deserted us. The ads from Floppy Disk Services, C.D.R., Studio Computers, Newline Software, Trionyx, The Software Toolworks, etc., are still there. Now, there are additional vendors supporting the Heath/Zenith community. You have ads from Dysan, HSC Inc., American Computer, Macrotech International, Hudson and Associates, D.E.L. Professional Systems, and other vendors of various products who would not have supported Heath/Zenith in years past when its user base was smaller.

The lesson to learn from all of this is to adapt to the current times. You obviously have learned the lesson, comparing the first issue of *Sextant* with the

current issue. The first issues dealt primarily with the H89 and HDOS. Your article content has changed to reflect your readers having upgraded and changed their systems. We must also learn to somewhat go with the flow, since if we all were too rigid to change with the advance of technology, everyone would be using H8s with 4K of memory and a cassette I/O system.

If Heath and Zenith settle their internal disputes, the Heath philosophy may again prevail in Heathkit and the Heath user community will live happily ever after. We, the end users, must be prepared to accept other outcomes such as Zenith divesting the Heath unit, or even worse, converting the Heathkit stores into a chain of stores like ComputerLand. If the latter does occur, I would hope that the third-party vendors like Trionyx, T.M.S.I., D-G, etc., who have thus far provided the enhancements we have wanted, will continue to, much like Trionyx taking the "orphaned" H8. I do not wish to be left in the position of today's Osborne owners.

On another topic, your response to a letter from Jordan Fox (page 6) took me aback. Having purchased my H89 three years ago from Heathkit and being quite satisfied with the service and support I have received from Heathkit and the Heath/Zenith community, including your publications, I have purchased a '150. Though I realize that some (apparently yourself included) would not consider this to be the "best" machine available, it serves me well and suits my needs. I have no regrets for not buying a '100 series. I am appalled that you apparently are deserting the H/Z150 users. If what you mean is that you do not wish to convert *Sextant* into a *PC World* clone, I support that decision. But, if what you mean is that the '150s will be ignored, then I suggest you are cutting yourself off from the fastest growing segment of the Heath/Zenith user community.

Come on now, Charlie, give us '150 owners some copy. I am not cancelling my subscription over this, but don't cut us off. Sometimes the choice of machines is dictated by the availability of software and not the machine.

William Flynn
Hopkins, MN

Telling it Straight

As a once satisfied Heath customer (*until* I bought my H100 in June 1983), I heartily applaud your editorial in the November-December *Sextant*.

It seems that the Heath/Zenith user community's biggest problem right now is Heath/Zenith. And HUG will hardly be of any help, since it's run by Heath/Zenith. (I've always felt that "Heath User's Group" was a misnomer—HUG is *not* a user's group, it's a company mouthpiece.) Clearly we need a user-operated national user's group/forum.

You mentioned HUG conferences put together by local groups. Are these groups run by users or Heath Company? (The Atlanta HUG is run by the local HEC manager and staff.)

You said that it is expected that ZDS will phase out the Heath Company. Do you mean with respect to computer sales, or will Heath go completely down the tubes?

I'll be *very* interested in what happens to the Heathkit Electronic Centers. I've never been in a store where the staff argues so much with the customers. I think if they were bought out by Toys 'R' Us, it would be an improvement. The trouble with the Heathkit Electronic Centers is that they hold out a promise of convenient local service that just doesn't wash. You're better off dealing with the factory in the first place.

Whatever happens, I just hope they'll get on with it. Either get their act together, or do the industry a favor and get out of the computer business. In the meantime, the Z100 owners, as well as the third-party vendors, are in a kind of limbo. Probably the best thing would be for users to make a point of buying from third parties instead of Heath/Zenith wherever possible.

The article on the history of the H89 was excellent. How about a similar article on the Z100? I've always been curious about who designed it, and why it wasn't made more compatible with the H89 (8085 vs. Z80, 2661 vs. 8250, etc.).

I'm happy to support you at the newsstand. Thanks for telling it straight.

John W. Spalding
Atlanta, GA

Various representatives of Heath Co. and Zenith Data Systems took advantage of regional HUG conferences held in November to deny that either the Heath Company or Heathkit Electronic Centers are being downgraded. They pointed to improvements being made to both. Their remarks will be covered in a future issue of Sextant.

Corrections

There are two errors in "Graphics Algorithm Optimized for 8080 or Z80," in the May-June 1984 *Sextant*. In the continuation of Listing 3 on page 75, the initialization statements:

```
X:= IX1;
Y:= IY1;
```

should be inserted after the line:

```
begin {LINE2}
```

Also, the fifth line down from there, which reads:

```
if IX2>=IY1
```

should read:

```
if IX2>=IX1
```

Thanks to Kenneth Jean of Detroit, Michigan, for pointing out these errors.

There was an error in one figure and an omission from the text in Robert Shakespeare's "Piggyback a Parallel Port on Your '89" (*Sextant* #13, November-December 1984).

Figure 2 (on page 68) showed the camera-ready artwork for the adapter board. Figure 2 should have appeared as it is in Figure A below. It was missing the entire foil that surrounds the circuitry and acts as a ground.

In the text, it should have been noted that two jumpers have to be placed between the circuitry and the foil ground. One jumper must go from pin 20 of the 8250. Another goes from pin 1 on the 6522.

A separate pad is provided for the 6522 jumper. The jumper itself can be seen in the upper-right corner of Photo 2 (page 73). A hole is drilled through the pad and another through the foil. The jumper is then soldered where it comes through the foil side. The jumper from pin 20 of the 8250 is simply tacked onto the foil with solder.

In Figure A here, you will see that pin 7 of the 74LS00 is connected directly to the foil. The foil and the two jumpers then serve to connect the 8250's pin 20, the 6522's pin 1, and the 74LS00's pin 7.

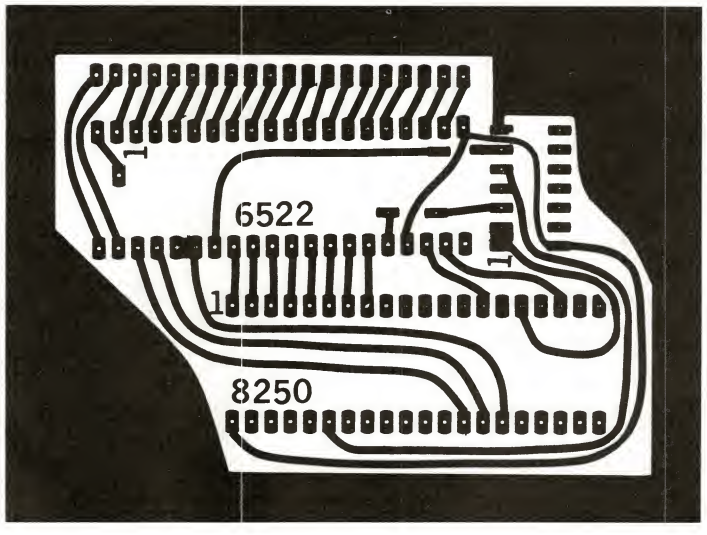


Figure A.

YOU'RE IN GOOD COMPANY

WITH AN H-1000

Amdahl, Nestles, Bell Labs, Jet Propulsion Laboratories, the U.S. Army and Navy, Allied, Bendix, CDR – these are just a few of the many satisfied H-1000 users. They know that the H-1000 gives them performance and compatibility unmatched by the giants. And many distinguished hobbyists have discovered that the H-1000 is available at a very attractive price – typically half the cost of a new IBM-compatible. Just look at the benefits:

- emulates IBM PC with monochrome board; runs most PC software at 4 times the speed
- 100% H89 compatible; runs your programs twice as fast with NO changes
- on-board memory from 128K to a full megabyte; holds twice the data and programs of any comparable PC!
- built-in Ghost disk for all operating systems; adds an additional RAM disk for instantaneous disk accesses.
- from $995 to upgrade your H89, or $1995 for a complete computer.

A STATE-OF-THE-ART UPGRADE

TMSI has taken the best features of Heath's 8-bit H89 and combined them with the best of the 16-bit PC's to give you the best of both worlds . . . and the ONLY package that builds on your current investment.

The 8-bit side gives you access to the world of low-cost 8-bit software for the CP/M and HDOS operating systems. The H-1000 runs all Heath/Zenith software and hardware for the H89 family without change. And with its 4MHz Z80, expanded memory, and ghost disk, your existing programs perform like never before.

The 16-bit side of the H-1000 features an 8086 running at over 8MHz, offering you the fastest screen response and program execution of any PC compatible on the market today. The standard keyboard and screen emulate an IBM PC with monochrome display; accessory color and graphics boards can extend the emulation even further. The "Ghost disk" is an added bonus; entire disks can be loaded into memory, freeing your "real" drives for another disk. You get instantaneous access to your data that cuts out all that waiting while your computer talks to your disk drive.

FULL SERVICE BACKUP DIRECT FROM THE FACTORY

The H-1000 comes fully assembled and tested, with Heath's "we won't let you fail" quality manuals. A full 90-day warranty covers each unit, with an optional service contract available. And we work closely with our customers for special applications assistance. Get the benefits of high-performance computing today! Call or write:

TMSI, Dept. RN
366 Cloverdale
Ann Arbor, MI 48105
(313) 994-0784

TMSI

Circle #173 on Reader Service Card

Microsoft Word and Mouse: Do They Click?

These well known IBM-compatible products may be better off apart.

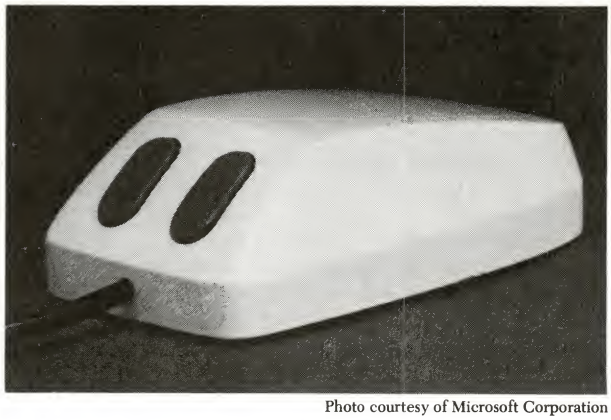


Photo courtesy of Microsoft Corporation

David R. Felstul

We recently received Microsoft's much-hyped word-processing program here at *Sextant*. It is called, appropriately enough, Word, and it is written for the IBM Personal Computer. This was one of the reasons for *Sextant*'s interest in Word. Among other things, we figured that Word would allow us to test our new Z150—Zenith's version of the IBM PC—to find out just how good a clone it was. Besides, with both Word and Microsoft's Mouse, which we received for use with Word, we now have many of the advantages that users of that *other computer* are reputed to have.

The Microsoft Word software shows just how faithful a clone the Z150 is. Although written for the IBM PC, Word works just fine on the Z150, which runs Microsoft's Disk Operating System (MS-DOS). Microsoft's Mouse, however, stole the show. It turned out to be lots of fun and quite useful in many ways, although we didn't find it as useful for word processing as its publicity led us to expect. (But I'm getting a bit ahead of myself. Let me start at the beginning.)

Conversational documentation

The first thing one is supposed to look at with new software is the instructions. Microsoft Word comes with a bit over 400 pages of documentation in a small, loose-leaf binder, along with a quick-reference card. The documentation is divided into three main parts.

The "Learning Word" section takes you, one command at a time, through a tutorial on how to write and edit text using Word. The next section, "Using Word," goes into depth on how to format text for printing with special characters or "fonts" such as italics. It also contains an explanation of the Windows feature, which allows you to split the monitor screen into several smaller screens or "windows." "Reference to Word," the last section, will be of more help to the experienced user than the other sections are. It contains an explanation of the concepts behind Word, the commands available in it, and messages that might appear while using it.

Thus, if you want to know what it means when your cursor turns × shaped, you can look in the "Elements" part of the reference section. The Elements part explains how to use the "elements" of Word: how to select text, choose commands, scroll the text, etc. There you'll find that the × means that Word is ready to cancel a command.

If, on the other hand, you want to know what the Print Queue command does, look in the Command Directory. (The Print Queue command lets you control printing from within Word, instead of from MS-DOS.)

Or if the message "Key Codes Conflict" appears on your screen, the Message Directory will tell you that two different font styles have been defined with the same key code—a two-letter code that you give to a font style of your own design. It's practical information like this that makes the reference section so useful for quick answers.

Other useful information is found in the Appendices at the end of the Microsoft Word documentation. The Appendices contain information on printing documents, converting WordStar files to Word files, and the like. A glossary and an index are also included in the Appendices.

Overall, Word's documentation is surprisingly non-technical. In fact, at times it is written in an almost conversational tone. Although this may make it more "friendly," the documentation tends to be disorganized and rambling. Particularly, I don't like the way that seemingly unrelated topics were lumped together into chapters. For instance, Chapter 5, Editing and Managing Documents, contains such diverse topics as Cutting and Pasting, Changing Disks, and Closing Windows. Thus, it is often hard to find exactly what you need without a lot of extra reading.

For the first-time user, I'd suggest diligently going through the Learning Word section first. Then skip the formatting section, Using Word, at least for the time being, in favor of hands-on experience. Don't be afraid to play around with Word. Its help command, ALT-h, is very useful if you get stuck. And in spite of its shortcomings, the manual does contain many helpful illustrations showing what your screen should look like.

Word's eight windows

One feature of Microsoft Word, Windows, is not commonly found on other word processors. This feature lets you create as many as eight small "windows." This lets you look at as many files as there are windows—all at the same time. The Windows concept is highly touted by Microsoft as an editing device.

I admit, eight windows does sound impressive. But I wonder how often anybody would ever use that many? Maybe once in a great while—if you're working on a really massive spreadsheet, for instance—but you certainly don't need that many for word processing.

Although another window or two might be used for referring to footnotes or a spreadsheet while editing text, it often seems easier to find other ways to do the same thing. Besides, Word displays only nineteen lines of text at a time as it is. Adding a lot of windows tends to decrease the already meagre amount of usable workspace.

One of the other characteristics of Word that other members of our staff weren't too crazy about was Word's repagination feature. You have to tell it to repaginate—it won't automatically do so when you add or delete material. As Frederick Zimmerman put it, "Word's repagination feature is clunky to say the least, especially compared to something like WordStar which has dynamic page breaking."

In addition, instead of something like WordStar's long lines of dashes separating pages, Word has a little arrow locat-

ed at the far left-hand side of the screen, next to the window border. This small arrow is easily overlooked, since it is only the size of a single character. To be fair, however, Word repaginates very quickly—even though it's hard to spot where it has put the page breaks.

Text-formatting capabilities

Word's powerful text-formatting capabilities are its greatest strength. Word can easily show you italics, justification, and indented paragraphs. It displays the formatted document directly on the screen—just as it will appear when printed on paper. (Assuming your printer can handle the formatting, that is.)

In other words, unless you have a laser printer or typesetting equipment that it supports, Word's ability to use 64 fonts with accuracies of better than one one-thousandth of an inch probably won't do you much good. If, for instance, you have a dot-matrix printer, trying to use Word's formatting capabilities may be like trying to put a Cadillac engine into a Volkswagen. Or it could be just the opposite—because Word doesn't seem able to take full advantage of all the impressive graphics capabilities of some dot-matrix printers.

Unlike the 64 different fonts, the idea of having a style sheet—an "indirect" way of formatting a document—seems like it would be useful in many different situations. Instead of storing print formats within the document itself, as is the case with direct formatting, the style sheet feature stores all the formats needed by the document in a separate file. Then when Word displays or prints the document, it looks at the attached style sheet file to see how to format it.

Style sheets can be useful if you want to write several drafts of a document.

Word also has an easy-to-use "undo" function—a good feature for those of us who are prone to hitting the delete key at inappropriate times.

You might, for example, want to print a double-spaced, single-column draft of your text for editing. All you have to do is type your text and then attach the appropriate style sheet. Word does the rest.

Now, suppose that after you've made all the necessary editing changes you decide that you want your masterpiece in a form suitable for publishing. That's just as easy. Once again, you simply attach the appropriate style sheet and Word will transform your deathless prose into a professional-looking article.

Another use of style sheets is to store frequently used formats. With a style sheet, you can save yourself the trouble of having to go through and define all the parameters of a format each time you use it.

Word comes with a couple of sample style sheets, and you can also create your own. A word of warning, however: it is a little tricky to create a style sheet of your own design and attach it to a document. It helps, for instance, to have a little understanding of typesetting terms before you start creating a style sheet. (This is where you should go back and read that section on formatting you skipped before.)

Cutting, pasting, and other basics

The ease of manipulating text by cutting and pasting is very important in a word processor since that is one of the biggest advantages a word processor has over a typewriter. With Word you can use the "Extnd Sel" (F6) key and function or cursor keys to extend the reverse-video highlighting to however many words, lines, or paragraphs you want to select.

Once cut, the text will go into the "scrap" buffer—the first and last few characters of which are displayed at the

"Quality Software at Reasonable Prices"

Software for Your H8/H89

HDOS Products

TEXT PROCESSOR 4.1 FULL SCREEN EDITOR	$ 59.95
TOUCH TYPIST 2.0	$ 29.95
VIDEO ARTIST PLUS, H19 GRAPHICS PROGRAM	$ 39.95
H25/Z25 GRAPHICS GENERATOR	$ 44.95
TOUR-700 GAME	$ 19.95
UNIVERSAL SPOOLER	$ 29.95
RESISTIVE NETWORK DESIGN	$ 24.95
MX-80 DEVICE DRIVER	$ 29.95
MODEM COMMUNICATIONS DEVICE DRIVER	$ 34.95
MBASIC COMPUTER-AIDED-INSTRUCTION	$ 59.95
HERO-1 ASM-68 6800 CROSS ASSEMBLER	$ 59.95
DENSE-PAC DATA COMPRESS AND ENCRYPT	$ 29.95
UD.DVD UNIVERSAL PRINTER DEVICE DRIVER	$ 29.95
MX.DVD MX-80 DEVICE DRIVER	$ 29.95
H25.DVD H25 DEVICE DRIVER	$ 29.95
UD.DVD/SPOOLER	$ 39.95
MX.DVD/SPOOLER	$ 39.95
H25.DVD/SPOOLER	$ 39.95
SUPER SYSMOD 2	$ 29.95
TINY PASCAL & DISASSEMBLER	$ 29.95
HFORTH-79	$ 34.95

CP/M Products

TEXT PROCESSOR 4.1 FULL SCREEN EDITOR	$ 59.95
TOUCH TYPIST 2.0	$ 29.95
TOUR-700 GAME	$ 19.95
DOMESTIC MGT. SYSTEM	$ 39.95
CHECK PROCESSOR	$ 29.95
TAXTYME 1984 FEDERAL INCOME TAX PROGRAM	**$ 59.95 NEW (1-15-85)**
SNAP	$ 29.95
MBASIC COMPUTER-AIDED-INSTRUCTION	$ 59.95
HERO-1 ASM68 6800 CROSS ASSEMBLER	$ 59.95
DENSE-PAC DATA COMPRESS AND ENCRYPT	$ 29.95
SUPERMATH 1 COMPUTER-AIDED-INSTRUCTION	$ 39.95
UD.COM/SPOOLER	$ 39.95
TINY PASCAL & DISASSEMBLER	$ 25.00
CNIX-80	**$ 65.00 NEW**
AUTOMOBILE EXPENSE	$ 29.95
APPOINTMENT CALENDAR	$ 29.95
H19/H89 BASIC GRAPHICS PRIMER	$ 49.95

Software for Your HERO-1

KEYPAD UTILITY CASSETTE	$ 24.95

HERO-1 ASM68 available for HDOS & CP/M

Software for Your Z-100

ZDOS Products

Z-100 HDOS 2.0 UPGRADE	**$ 79.95 SPECIAL**
SAVE $60 . . . . includes FREE Z-100 HDOS Text Processor	
TEXT PROCESSOR 4.1 FULL SCREEN EDITOR	$ 59.95
TOUCH TYPIST 2.0	$ 29.95
CHECK PROCESSOR	$ 29.95
SUPERMATH 1 COMPUTER-AIDED-INSTRUCTION	$ 49.95
AUTOMOBILE EXPENSE	$ 29.95
SOFTPLOT	**$ 99.00 NEW**

PROFESSIONAL TEXT PROCESSOR—NEW

For Z-DOS	$ 99.00
For CP/M-86	$ 99.00
For Z150 MS-DOS	$ 149.00

CP/M-85 Products

TEXT PROCESSOR 4.1 FULL SCREEN EDITOR	$ 59.95
TOUCH TYPIST 2.0	$ 29.95

Newline's Z-100 HDOS 2.0 Products

TEXT PROCESSOR 4.1 FULL SCREEN EDITOR	$ 59.95
UD.DVD UNIVERSAL DEVICE DRIVER	$ 29.95
MX.DVD MX-80 DEVICE DRIVER	$ 29.95
H25.DVD H25 DEVICE DRIVER	$ 29.95
UD.DVD/SPOOLER	$ 39.95
MX.DVD/SPOOLER	$ 39.95
H25.DVD/SPOOLER	$ 39.95
SUPER SYSMOD 2	$ 29.95

Specials

TEXT PROCESSOR 4.1 for ZDOS & CP/M-85	**$ 99.95 SPECIAL**
Z-100 HDOS 2.0 with FREE Z-100 HDOS Text Processor	**$ 79.95 SPECIAL**

Diskettes

SSSD, 10 sector Hard	10 for $25.00, 20 for $45.00
SSDD, soft	10 for $25.00, 20 for $45.00
DSDD, soft, 48 TPI	10 for $33.50, 20 for $62.00
DSDD, soft, 96 TPI	10 for $42.00, 20 for $79.00

CALL FOR QUANTITY PRICING.

Newline Software

Send for latest catalogue

P.O. Box 289 • Tiverton RI 02878 401-624-3322

ORDER FORM

FOREIGN ORDERS—Surface postage included in price. Add $3.00 per package ordered for airmail shipment. Checks drawn on a non-U.S.A. bank must include $10.00 for check processing and funds transfer.

CANADA ORDERS—Checks drawn on a non-U.S.A. bank must include $3.00 for check processing and funds transfer.

Mail to: Newline Software
P.O. Box 289
Tiverton, RI 02878

OS = Operating System—Enter either HDOS, CP/M, Z-DOS, or CP/M-85. Be sure the product you are ordering is available for the OS you select.

Format = 5.25″ Hard Sector (H17 Format) . . . use format code ''H''.
5.25″ Soft Sector (H37 Format) SSDD, 48TPI . . . use format code ''S''.
8″ SSSD (H47) Format . . . add $5.00 . . . use format code ''8''.
Z100 Software is in DSDD, 48TPI format. . . use format code ''Z''.

Quantity	Product Description	OS	Format	Each	Extended

S11-84

Rhode Island residents must include 6% Rhode Island Sales Tax . . . .

SUB TOTAL _____

RI SALES TAX 6% _____

TOTAL _____

☐ Please send latest product catalogue.

NAME _____

STREET _____

CITY _____ STATE _____ ZIP _____

Circle #145 on Reader Service Card

bottom of the screen. It's easy enough then to insert the scrap into the text by pushing the INSert key. However, it would be nice if you could edit or at least fully display the scrap instead of only a few characters.

I also didn't like the way the search command worked. The search command is limited to the first occurrence of the search string. A "next" key should be provided so you can search through a document for all the occurrences of that string. Then you could search for a word, change it in whatever way you wish—making it boldface, for instance—and then go on to the next occurrence.

You can't get around this barrier by setting up a macro function, either. In fact, you can't set up any macros, nor can you define function keys as you can in some other word processors. This is because Word is a screen-oriented editor and you don't type in commands—you just point to them in a menu. (A macro, after all, is just a file of previously entered commands that the program executes as though you were typing them in yourself.)

If you have never used macros, you might never miss them. However, I have made extensive use of those features on WatchWord, the screen- and line-oriented word-processing program we use on the Z110; and I have become rather accustomed to the convenience they provide. Macros and user-definable keys are two of the easiest ways to "customize" your word-processing software to fit your needs.

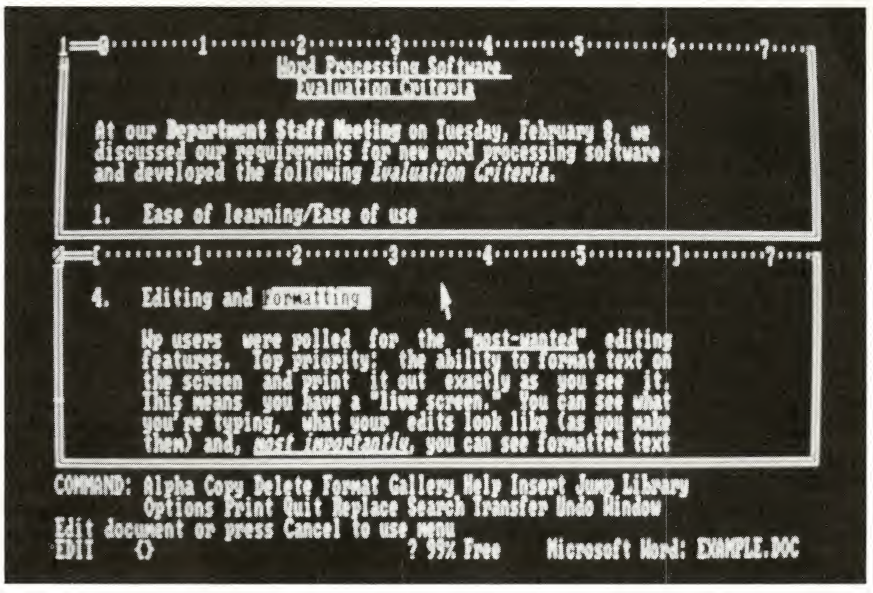


Photo courtesy of Microsoft Corporation

The Windows feature of Word lets you look at as many as eight different files, or sections of a file, at one time.

"Thumbing" through Word

Another problem with Word is its scrolling capability—or lack thereof. I had assumed that you could just race the mouse across a desktop and scroll up, say, a hundred lines or so. This isn't the way it works, however.

When using the mouse to scroll up or down in a document, you must move the cursor all the way to the left-hand window margin to the "scroll bar." There the cursor will change into a two-headed arrow pointing up and down. Clicking the left button will bring the line of text at the top of the screen down to the arrow's position—in effect, scrolling "up" towards the top of the document. Clicking the right button will send the line of text at the arrow up to the top of the screen—in effect, scrolling "down" towards the bottom of the document. In both cases Word then fills in the rest of the text as needed. (This is not the way it's explained in the manual—but this seems to be closer to the way that Word really operates and thus, perhaps, is easier to understand.)

Word's means of scrolling with the mouse is both confusing and annoying; and I found it to be one of Word's most poorly designed features.

There is a quicker way to move around in a document with the mouse—but it, too, is more difficult than it should be. You can do what Word calls "thumbing" (the way you'd thumb through a book). To do this, you move the cursor to the scroll bar and press *both* buttons. If the cursor arrow was in the middle of the scroll bar—halfway between the top and bottom of the window—the screen will scroll to the middle of the document. If the cursor was located three-quarters of the way down the scroll bar when you pushed both buttons, Word will scroll to a spot three-quarters of the way through the document; and so on.

Word's "thumbing" seems to have the same disadvantage as its real-life counterpart—it's very inaccurate. You often go right past the page you want. And you can't thumb to a place in the document with the mouse and then use the cursor controls (the arrows) on the keyboard—because the cursor is still sitting wherever you thumbed from. The cursor doesn't follow the thumbing—you have to use the mouse to relocate it. Compare this with the keyboard method of moving the cursor. Using Word's function keys, a single keystroke can take you up or down, quite accurately, one page at a time.

Other features of Word

Word also has some well designed features. To begin with, unlike scrolling or thumbing with the mouse, Word's "glossary key" (F3) does make things quicker and easier. It will insert a long phrase, previously defined by you, into your text when given a glossary name of two letters or more followed by the F3 key. This can come in very handy with long phrases that you use often. It's somewhat analogous to typing a word and having a dictionary automatically supply you with its definition, hence the name "glossary." Word can store hundreds of such "definitions" in its glossary.

Word also has an easy-to-use "undo" function—a good feature for those of us who are prone to hitting the delete key at inappropriate times. By typing ESC U, you can undo your last editing change. (I've probably become more familiar with this key than I'd care to admit.)

And finally, perhaps Word's best feature is its use of detailed menus—similar to the menus used in Microsoft's Multiplan spreadsheet. Complete and easy to understand, they are made even simpler to use by the addition of the Microsoft Mouse—just point and click.

Introducing a very talented rodent

As I mentioned above, *Sextant* received the Microsoft Mouse along with Word. The mouse is designed to let you move around the screen and make selections from the menus more easily than you can by using the keyboard's spacebar and TAB key. Microsoft's Mouse thus provides many of the same advantages as its more famous relative, the mouse that comes with Apple Computer's Macintosh and Lisa. It differs in one important aspect, however. It has two buttons you can push, compared to Apple's one.

This extra button presents many more choices and consequently many more chances for the user to make errors. You can click the left button, the right one, or both. You can hold the buttons down for a while and then release. You can even press the left mouse button in concert with the ALT, CTRL, or SHIFT keys on the keyboard.

Of course, each combination of the buttons does something completely different. This can be hazardous to your document's health since it is difficult to remember which button combination does what. Your memory is not aided

very much by the cursor icon, either. It can turn into any one of over a dozen different shapes, depending on the button choice and where the cursor is. Although each cursor shape has its own particular meaning, there are just too many shapes to remember.

If the mouse is supposed to simplify things, why so many choices? Why not have just one button and use an additional menu or two? This would be easier than remembering things like the "mouse method" of selecting the entire document for an editing change. To do that, you have to move the cursor over next to, but not on, the scroll bar along the left window border. At that point, you're on the "selection bar" and the cursor arrow shifts to the right. You then click both mouse buttons at the same time to select the whole document.

Once again, compare this with the keyboard means of selecting the entire document: just hit the SHIFT-F10 key. You have to use the keyboard for typing in text anyway. So even though the mouse often simplifies the use of Word's menus, I often just park it and use the keyboard for everything—rather than shifting back and forth from mouse to keyboard.

Some of *Sextant*'s other staff members have also found it easier to work this way. They thought that the mouse demanded too much manual dexterity to operate, especially in word processing, to make it worthwhile to use it. Selecting a single text character for editing requires a very deft touch when using the mouse. So instead of using the mouse, many people may find it easier just to use the cursor controls on the keyboard.

A PIANO-playing Mouse

Microsoft supplies three programs with Mouse to show off its capabilities: PIANO, LIFE, and DOODLE.

In PIANO, you use the mouse's cursor to point to piano keys. The PIANO program will then sound the appropriate note. It's fun, but it also requires more manual dexterity than you might expect. (In other words, you'll never be able to play the Maple Leaf Rag with PIANO. A recognizable rendition of Three Blind Mice, however, is not beyond the realm of possibility—and it does seem more appropriate somehow.)

The second program, LIFE, consists of a 20- by 39-line grid, each square of which represents the possible location of a biological cell. The rules of LIFE are simple: If an existing cell had one or no neighbors in the preceding generation it dies from isolation. Two or three neighbors—it survives. Four or more neighbors in the previous generation and it dies from overcrowding. And if an empty square had exactly three neighbors before, a new, living cell is born in that square.

The simulation starts after you create a preliminary arrangement of living cells using the mouse cursor. It progresses through as many generations as you want, with the computer deciding whether each cell lives or dies—according to the rules.

Many arrangements of cells go through interesting patterns and it's fun to discover which ones do so. You can also choose from a menu how you want LIFE to run. It can run either one generation at a time or continuously (one generation right after another). Be forewarned, LIFE is quite addictive; so plan your time accordingly.

The third program that comes with the mouse is DOODLE. It lets you do exactly what its name says. It comes with its own separate nine-page booklet of explanation. You can really get into some high-tech doodling with DOODLE by drawing exotic patterns and filling them in with vivid colors.

All three of these programs are, admittedly, very enjoyable. But they also have nobler motives. PIANO helps you develop your dexterity with the mouse. LIFE gets you started using menus. And DOODLE demonstrates the graphics capabilities of the mouse. The programs are so much fun that you don't even realize that you are learning something which may prove useful later on.

Of mice and menus

Microsoft also provides some prewritten mouse menus so that their mouse can be used with some popular data- and word-processing programs, such as Lotus 1-2-3, Multiplan, VisiCalc, and WordStar. This capability demonstrates just how versatile the mouse is. The preset menus are explained in the 60-page Microsoft Mouse Menu booklet—along with directions for creating your own mouse menu.

For more adventurous types, there's even a mouse programming language to let you use the mouse in your own programs—for setting up menus or playing games, for example. To help you out, the documentation includes a listing in BASIC of the PIANO program. After studying the PIANO listing and the list of mouse system calls in the documentation, I wrote a simple program in BASIC to try out the mouse in this fashion.

My program set up three boxes towards the left-hand side of the screen and an elongated one at the bottom. Each of the three boxes contained a foreground and background color scheme for the screen. The bottom box was the "quit" box.

Simply put, my menu-like program allowed me to use a mouse cursor of my own design to point to the color scheme that I wanted. And, by clicking the left button, I could change the screen to those colors. (The program would send the appropriate command to the screen.) If I wanted to quit, I just moved

INTERROGATOR™

Drive Diagnostic Program for the ZENITH Z/H-100 Series

Diagnosing disc drive problems has just become a great deal easier.

Interrogator,™ Dysan's economical new drive diagnostics package, can run a drive through its pace in minutes. To tell you exactly what's wrong. Or right.

Just like that.

Test results come right up on the screen of the machine you're testing. Or you can print out hard copy records.

The Interrogator package includes a Digital Diagnostic Diskette,™ complete test instructions and software programmed to perform a whole battery of sophisticated tests.

Things like head radial alignment. Head positioner linearity. Head azimuth alignment. Index timing. Spindle speed. Read/write verification. And over a dozen others.

Interrogator is here now for the Zenith Z/H 100 Series. Testing both 5¼" and 8" models. Versions for other computer models are on the way.

Whether you're going for one drive or need to assure the compatibility of a company full of drives, Interrogator is the perfect tool.

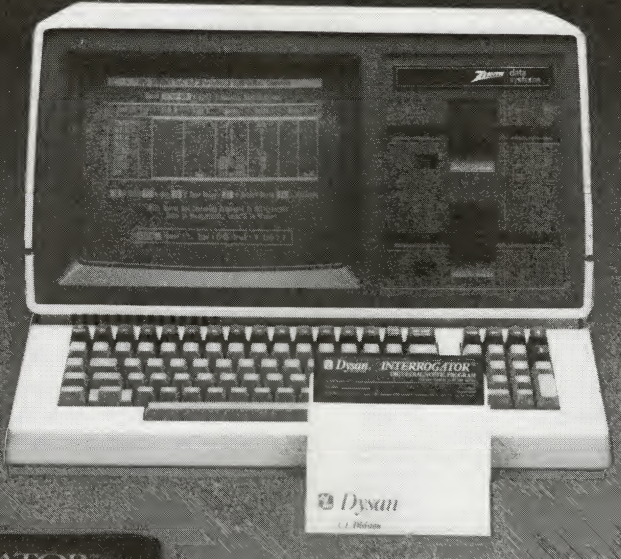
And it's just the kind of thing you'd expect from Dysan. The leader in quality, innovative media and diagnostic tools.

Interrogator. What could be easier?

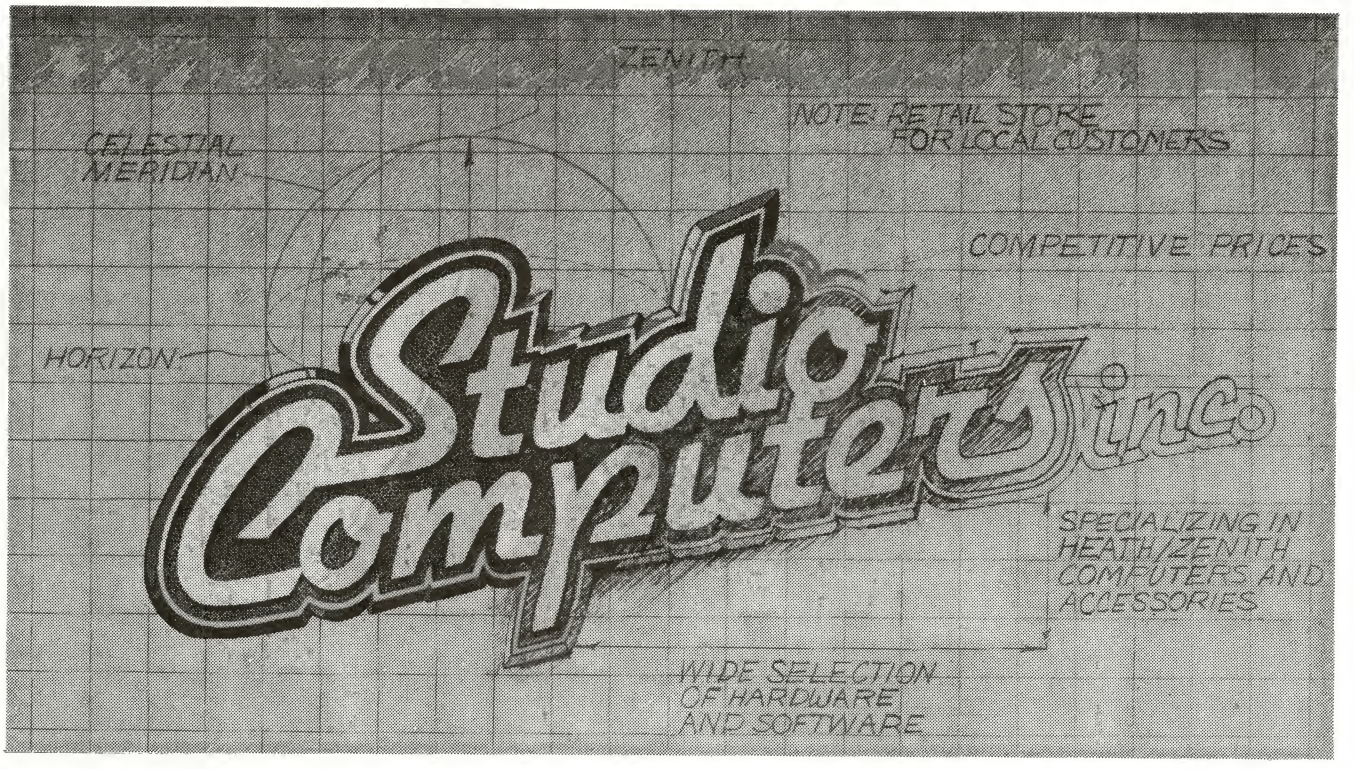
Just this. A toll free phone call to get more information. Or place an order. (800) 551-9000.

Dysan®

5201 Patrick Henry Drive
P.O. Box 58053
Santa Clara, CA 95050
(408) 988-3472



Circle #143 on Reader Service Card



DESIGNED TO BRING YOU THE BEST IN PRICE AND PERFORMANCE

HERE ARE JUST A FEW OF THE ITEMS WE CARRY...

Z150 PRODUCTS

PRO DRIVER Communications Ver 2	$49
dBASE III Database Manager	$525
FRAMEWORK Integrated Software	$525
SYMPHONY Integrated Spreadsheet	$525
DOLLARS & SENSE Home Finance	$155
SPINNAKER Children's Software	$Call
SUBLOGIC Night Mission Pinball	$35
BLUE CHIP MILLIONAIRE Game	$49
BLUE CHIP BARRON Game	$48
PRO KEY Macro Key Utility	$99
SIDEWAYS Printer Software	$49
DESIGNWARE Children's Software	$Call
MICROSOFT MOUSE and Software	$175
KEYTRONICS 5151 Keyboard/cursor	$215
WINCHESTER Hard Disk Subsystems	$Call
SMARTWORK PC Board CAD Software	$Call
TECMAR Captain Multifunction Board	$335
AST SIX-PAK + Multifunction Board	$335
TECMAR TIMEMASTER Clock/Calendar	$125
IOMEGA BERNOULLI BOX 10 & 20 Meg	$Call

ZENITH DATA SYSTEM

Call us for Super Z100 & Z150/160 PC prices, as well as accessory hardware and software.

MISCELLANEOUS ITEMS

UCI 256/512K RAM Board for Z100	$655
SILVER REED 550 Daisy Wheel Printer	$545
SMARTCABLE AutoLogic RS-232 Cable	$75
MITSUBISHI M4853 96TPI DS/DD Drive	$225
DATAPRODUCTS 8050 Color Printer	$1516
DATAPRODUCTS 8010 B&W Printer	$519
ACCUTRAK 48TPI DS/DD Disks	$27
TANDON TM-848-2E 8" Drives	$419
MPI PRINTMATE 99G Printer	$Call
MPI PRINTMATE 150-A1 with Keypad	$Call
MPI "X" SPRINTER Printer	$Call
MPI AP-PAK Software Packages	$59
NOVATION SMARTCAT+ 300/1200 Modem	$375

ZENITH data systems

999 South Adams
Birmingham, MI 48011
Phone (313)-645-5365

Please add 2% of the total order (minimum $2) for shipping, insurance and handling for anywhere in the continental U.S.

VISA MasterCard

For a free copy of our 24 page illustrated catalog describing all our products, use the reader service card or call us anytime from 9:30 to 5:30 Eastern time.

Circle #249 on Reader Service Card

the cursor to the "quit" box and, once again, clicked the left button.

Another option I put in my program allowed me to click the right mouse button and no matter where the cursor was on the screen, the screen returned to its original colors, black and white. (Clicking both buttons, by the way, had no effect anywhere in my program.)

Learning how to call the mouse from a BASIC program turned out to be relatively easy. The Mouse Installation and Operation Manual provides 123 pages of clear, concise directions on how the mouse works. It tells you how to modify Mouse parameters like horizontal and vertical sensitivity (measured in a unit called a "Mickey"), by using mouse system calls.

The BASIC PIANO listing in Appendix B—which includes a summary of what the parts of the program do—helps immeasurably when creating your own programs. You can even lift whole routines out of the PIANO program to use in your own programs.

Appendix C is almost as helpful. It shows eight different mouse cursor shapes you can use—I chose the pointing hand, for example. It even lists the BASIC routines that produce them so you don't have to know their binary code in order to use them.

Are Word and Mouse for you?

In spite of some very powerful capabilities, Microsoft Word is certainly not for everyone. It's only an average word processor. But if you need some fancy formatting, Word may be just the thing for you. (So long as your printer can reproduce all those different fonts, that is.)

Companies or individuals, for instance, who produce a lot of their own final copy might find that Word's extensive formatting abilities mesh very well with their needs. Text formatting can effectively emphasize the most important parts of a memo, or dress up an advertising circular or press announcement. But the computer owner with a dot-matrix printer or little need for classy correspondence might be advised to look elsewhere, especially considering Microsoft Word's price—$375 ($475 with the Mouse).

You simply attach the appropriate style sheet and Word will transform your deathless prose into a professional-looking article.

On the other hand, Microsoft's Mouse must be one of the most adaptable hardware enhancements you can buy—if you have the dexterity to use it to its fullest. It isn't essential for word processing; but if you feel more comfortable using a mouse, you might consider it for that purpose. Or if you write a lot of your own programs (especially if you plan on selling them), consider the mouse as a valuable way to increase your program's appeal and ease of use. Even if you just plain like to tinker with computer enhancements, Microsoft's Mouse will provide you with hours of enjoyment. (The Microsoft Mouse comes with the Mouse Menus for Lotus 1-2-3 and other software programs mentioned earlier.)

Ordering Information

Microsoft Word (without Mouse), $375.
Heath Cat. #MS-5063-9
Heath Company
Benton Harbor, MI 49022
800/253-0570 for credit card orders

Microsoft Mouse, $195.
For location of nearest dealer contact
Microsoft Corporation
10700 Northup Way
Bellevue, WA 98004
800/426–9400
(206/828–8088 from Washington)

MEMO:

to: H/Z-100 users

Thank you for your overwhelming response to the P-SST card! Our first two editions are SOLD OUT! However, new production is available as you read this, so don't hesitate to order!

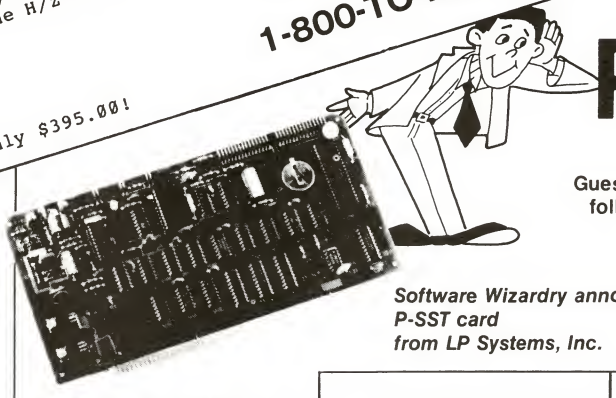
Don't forget that a disk-full of software comes with the P-SST, including graphics/sound and speech demos, and three-voice musical scores. To encourage more software support, we are now selling the P-SST Development Libraries, which make programming the P-SST amazingly easy!

The P-SST's functions are also creatively utilized by our applications programs CHRONOLOGIC (CLOCK/CALENDAR), PALETTE GRAPHICS (JOYSTICK SUPPORT), REACTOR-100 NUCLEAR REACTOR SIMULATION (SOUND/VOICE), AND ZLYNK/II MODEM COMMUNICATIONS (CLOCK/CALENDAR). We think you'll agree the Programmable Sound/Speech/Time card is one of the finest support products yet available for the H/Z-100!

Sincerely,
Software Wizardry, Inc.

P.S. The price is still only $395.00!

(314) 946-1968
1-800-TO-BUY-IT



PSST!

Guess what sings, hums, talks, follows your every move, and plugs into your Z-100?

Software Wizardry announces the P-SST card from LP Systems, Inc.

Phoneme Speech Synthesizer — Programmable features include a directory of 64 phonemes, with four pitch levels.

Joystick Interface — Two general purpose 8-bit parallel I/O ports. Software included to configure them for use with standard Atari joysticks.

IEEE-696 (S-100) bus — Designed for the Z-100 computer, the board complies completely with the bus standard and can be interfaced into any S-100 system.

Audio Power Amp — On-board power amplifier provides one watt of output power to drive standard 8 ohm audio speaker.

Audio preamp out — Provides low level audio signal out, suitable for connection to a standard preamplifier input.

Sound Synthesizer — Provides three channels of music/sound and one channel of white noise generation, independently programmable for simultaneous operation. Amplitude can be under direct program control or controlled by envelope generator. Control registers remain latched to most recent instruction to reduce CPU overhead.

Frequency generator program listing provided, and sample sound and music programs.

Clock/calendar — On-board clock with 9-month lithium battery always knows the correct time and date. With the CHRONOLOGIC program, on boot- up ZDOS will always know the time. ALso can be programmed to provide elapsed time; Alarm interrupt at pre-set time and "heartbeat" repetitive interrupt at specified intervals to S-100 interrupt bus. "Standby" interrupt provides external TTL signal to control an external device, such as an relay to allow your Z-100 to turn itself on!

We admit it doesn't dance, but you might after you plug it in!

The Programmable-Speech/ sound/ time card is the latest result of Software Wizardry's crusade to fill the holes in your Z-100!

Not just a piece of hardware for programmers and system integrators, we're supporting it with SOFTWARE; we don't leave you out in the cold.

In addition to the standard utilities that come with the card, the clock read function is supported by our CHRONOLOGIC program; The joystick interface is used by our best selling (and still the best!) graphics drawing program PALETTE, (which fills another hole with its light-pen support), and appropriate sound effects issue from it when you use it with our REACTOR-100 nuclear reactor simulation.

And thats just the start; how about that for support! We know you all will be quick to make use of P-SST, because we love the Z-100, and we know you do too!

Try our card in your computer, and PSST!!!, pass it on!

Available direct from Software Wizardry, Inc. as well as from most Heathkit Electronic Centers and many Zenith Data Systems dealers and distributors as well. Please add $2 minimum or 2%, whichever is greater, shipping/handling plus sales tax if shipped to a Missouri address.

$395!

Software Wizardry, Inc. also has over 450 other items of interest to Heath/Zenith users as well. Call or write for our FREE price list. Dealer inquiries and HUG group purchases welcome!

MasterCard VISA

ZENITH data systems

Software Wizardry

1106 First Capitol
St. Charles, MO 63301
(314) 946-1968

Circle #221 on Reader Service Card

TMSI's H-1000: Grafting an 8086 Onto Your '89

Now you can have dual processors—and hang on to your '89. (But it's still not a Z100.)

Wayne Rash, Jr.

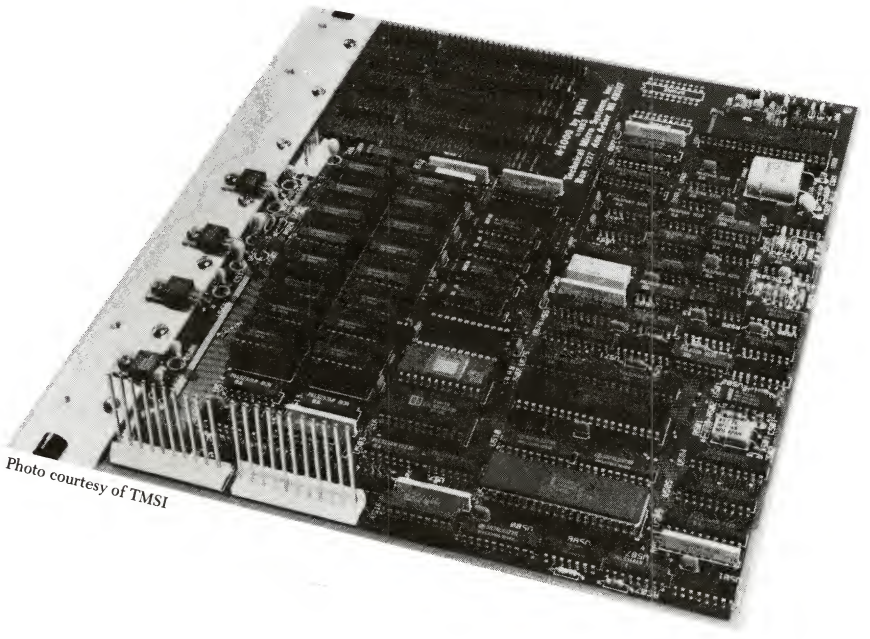


Photo courtesy of TMSI

It's an H89 version of the Q-ship—high-powered, heavily armed, and masquerading as a harmless steamer.

On the outside, there's little indication of the power that lurks within the apparently mild-mannered computer. It looks just like any other H/Z89—until you realize that this one runs at a blistering 8-megahertz pace, backed up with 256 kilobytes of random-access memory (RAM). Then there's the lettering on the screen that looks a little odd—what's this about "IBM Personal Computer DOS"? On an '89?

In this case, a computer that started out life on the shores of Lake Michigan as a stock H89 has become a 16-bit powerhouse that is a great deal more than it looks to be. It has not forgotten its heritage, however. As it always did, it still runs the Heath Disk Operating System (HDOS) and Heath/Zenith CP/M. The same accessory cards still plug into the expansion locations (with some limitations). Software written for the '89 runs as always.

The product that gives the '89 this dual personality is the H-1000 from Technical Micro Systems, Inc. (TMSI) of Ann Arbor, Michigan. The H-1000 replaces the central-processing unit (CPU) board in an H/Z89 computer. This new board brings with it an Intel 8086 processor, up to a megabyte of main memory, and the 8-MHz clock speed mentioned earlier. You still get the Z80-based system that you are used to for the 8-bit side of the computer, thus allowing your operating systems and software to run as you expect them to. You can run at 4 MHz or at the '89's usual 2 MHz. It's easy to switch between speeds in case you have software that requires the slower speed.

As you would expect, you get some additional operating systems and software for the 16-bit processor to use. When you order the H-1000, you can choose between Digital Research's CP/M-86 and the Microsoft Disk Operating System (MS-DOS). In this case, MS-DOS is provided in the form of IBM's PC-DOS.

This machine gives a great first impression. Suddenly you have four operating systems available where you once had two. You can run PC-DOS. You begin to feel that you have the ultimate solution to computing. Regrettably, you do not. None of this ability comes without cost, and while the costs involved are not unreasonable, they must be considered.

The most obvious cost is the $1,495 you must pay for the basic 256K CPU board with one operating system. This may be more than you paid for the '89 you are upgrading. If you do not own a soft-sector disk controller or 8" disk-drive controller, you will want to add that, too. This will add some additional dollars to your expenditure. Depending on your situation, you may find some additional costs as well. More on those later.

What you get

The H-1000 is available in several different versions. The most common is a CPU board that is designed to replace the CPU board already in your '89. There are also versions to upgrade H/Z19, H/Z29 and H/Z49 terminals into full computers.

The '89 version, which I tested, comes fully populated with all components installed except for some read-only memory (ROM) chips. You must take the ROMs out of the '89 CPU board you are replacing. If you ordered more than the standard 256K RAM, it will already be installed on the board for you.

You also receive a manual from TMSI explaining the installation and use of the H-1000 board. And you get some additional parts required to perform the installation. A diagnostic program comes with the package as well. Therefore, you can test your '89 before installation begins, and again after it is complete. Your operating system and any additional software you ordered is packaged and shipped separately.

Installing the H-1000

If you built your '89, you're not going to have any trouble installing the H-1000 circuit board. The manual takes you through the process step by step, and is organized very much like a Heathkit assembly manual. The primary stages of installation include upgrading the power supply, changing a plug, setting jumpers, installing the accessory circuit boards, and placing the H-1000 into the '89.

Not all '89s require that the power-supply upgrade be made. Later models with the black power-supply heat sinks

probably will be adequate. If you have an earlier version, you must install new heat sinks and a new voltage regulator to handle the requirements of the H-1000. Also, you may have some lines going through an in-line plug that will have to be removed and soldered to the power-supply board instead. This last is a Heath-recommended change, by the way. Those plugs have caused some problems through overheating.

The power-supply upgrade requires that you remove the supply from the computer, make the changes, and replace it. This operation is not difficult, and it is explained in detail in the instructions. You are also told how to identify the changes you must make to your particular power supply.

After you get the power supply back into the computer, the next step is to change the plug shell that connects to the CPU-board power connectors. Again, which operation you perform depends on the version of '89 you have. Older models call for swapping the 10-pin shell that connects to location P514 for one with 11 pins. If you have a newer model, you must swap the 11-pin shell that connects to location P515 for one with 10 pins. This sounds confusing, and it is, but the manual is very clear in this area. If you follow it through one step at a time, there should be no problem in making the switch.

Deciding which change to make is more difficult than actually making the change. The H-1000 kit includes the proper connectors; so all you have to do is follow the directions step by step. You are told which wire to remove from the old plug shell, and where to insert it in the new shell. You are also given a chart that shows exactly where each wire is to go.

Changing the plugs is probably the most difficult part of the whole process, which gives you an idea how easy the process is. Setting the board's configuration is very straightforward. You are shown a drawing of the CPU board that shows the location of each of the jumpers. The proper setting for each of them is explained. You will need to have your old CPU board handy for this, incidentally, since some of the explanations are made in terms of jumper positions on the old board. At this time as well, you will set switch SW501 on the CPU board for your particular system configuration. The settings will be the same as they were on your old board.

The last step before trying out your new CPU board is the ROM swap. You must remove the three or four ROM chips from the old CPU board and install them in the new board in their corresponding locations. The new locations are numbered just as they were on the original board, so this is not difficult. Again, the manual gives you a drawing of the locations on the old and new CPU boards. You are also shown exactly how to remove and replace an integrated circuit (IC) chip. An IC-extraction tool is included with the H-1000 kit.

Now comes the big moment. You slide the new CPU board into place. You attach the power connectors and the connectors from the terminal logic board. You turn on the power, hear the two beeps, and see the H: prompt on the screen. The new board works. You only have to install the accessory boards to complete your project.

The H-1000 has five expansion locations that correspond to the three right-hand expansion locations on the '89. The left-hand expansion locations are not carried over to the H-1000. Since nearly every expansion board for the '89 uses the right-hand locations, you can gain some flexibility here.

You should install any accessory boards from the '89 onto the corresponding locations in the H-1000. Since the mounting locations on the H-1000 are slightly farther to the right, you may have to reroute a disk-drive cable. Once you have completed this, you are ready to put everything back together and start using your upgraded '89.

The H-1000 is designed to operate as an '89 and to run all '89 software.

Using the H-1000

The first thing you should do with your H-1000 is run the diagnostic program that came with it. This is the same program that you ran to make sure your '89 was running okay before you installed the H-1000. Now you need to confirm that all is still well. Some non-Heath accessories may cause problems with the diagnostic program; but you are told about that ahead of time.

When you are done running the diagnostics, you're ready to try out your new 16-bit '89. Locate your copy of CP/M-86 or MS-DOS. If you are using CP/M-86, place the system disk into drive A: and press B to boot the system. You will see the Digital Research sign-on message in a few seconds, and your system is running on 16 bits.

Starting up MS-DOS is a little more complicated. Since TMSI actually sells IBM's PC-DOS 2.0 as its version of MS-DOS, you must first fool the machine into thinking it's an IBM instead of an H-1000. You do this by booting with a ROM emulator, and then booting PC-DOS. As you may have guessed from this, it requires that you use two disks each time you want to use PC-DOS. You boot first with the ROM-emulator disk, and then swap disks and boot again with the PC-DOS disk. When the second disk loads, you will see the "IBM Personal Computer DOS" message appear on the screen of your '89. For those who are so inclined, Zenith's MS-DOS 2.1 for the H/Z150 also works fine. At least this way you can stay "pure Heath/Zenith" in your operations. But you will still need to boot with the ROM emulator.

A significant enhancement to the operation of the H-1000 is the inclusion of a "Ghost Disk" utilities package as a part of the software you receive. This allows you to use part of the H-1000's RAM as a solid-state disk emulator. This utility is present for both of the operating systems supplied by TMSI.

Since the solid-state "Ghost" disk emulator treats a section of the computer's RAM as a very fast disk drive, the operation of some programs can be speeded up substantially. This is particularly noticeable with programs like Ashton-Tate's Friday! which make very heavy use of the disk drives. TMSI also includes a "Ghost Disk" utility for Heath/Zenith CP/M-80 and for HDOS. This way, you can gain a great deal of benefit from that 256K RAM even with 8-bit operating systems.

There is another enhancement for the 8-bit side of the machine as well. While the H-1000 will operate at the normal 2-MHz clock rate of the '89, it is also capable of running at 4 MHz. You change speeds by running two programs, one called "Fast," to speed things up; and the other called "Slow," to slow the computer back down to 2 MHz. I was pleasantly surprised at the difference brought about by speeding up the computer.

In most other respects, the H-1000 is pretty ordinary. This is precisely as it should be. The H-1000 is designed to operate as an '89 and to run all '89 software. This it does. In the 8-bit mode, it will run nearly all '89 accessory boards as well. About the only exception to this is the 16K memory-expansion board, which is not necessary.

The very fact that it appears so ordinary is one of the H-1000's strong points, for you can be assured that you will not need to convert any of your present software.

Problems

When you consider the nature of the modifications involved, the H-1000 has surprisingly few problems. None of the problems I found had any relationship to standard Heath/Zenith software or hardware. In addition, most non-Heath hardware works fine, too.

The only exception I found in hardware was the Magnolia Microsystems soft-sector disk controller. According to TMSI, this accessory board will work fine for 8-bit operation; but it will not

Make Your Heath/Zenith System More Complete—Subscribe to *Sextant* Today!

SEXTANT

The Independent Magazine for Users of Heath/Zenith Microcomputers

Whether you use an **H/Z100, H/Z89, Z90, H88, or H8**—you'll find articles in every issue which apply to your system.

Explore **CP/M, HDOS, and Z-DOS** capabilities in *Sextant*. Applications, programs, compatible hardware and software, and the latest developments in the Heath/Zenith community are among the topics you'll find in every issue of *Sextant*.

Regular *Sextant* features include:

- **"How-to" articles** to help you enhance your system. Read how other users have altered their systems to suit their needs—and how you can too.
- **Reviews of products** from Zenith and independent suppliers, which can help answer your questions about hardware and software purchases you're considering.
- Short program listings—including **utility programs and games**—which you can use immediately.
- **Advertising by independent suppliers** who support Heath/Zenith systems. In *Sextant* you'll find many products advertised that you just won't find anywhere else.
- Coverage of community affairs. You'll read about **major events and personalities** in the Heath/Zenith user community, and what effect they could have on you.

You can't afford to miss all the information packed into each issue of *Sextant*. **Your Heath/Zenith system just isn't complete without a *Sextant* subscription!**

Get the information you need to get the most from your system!

Start your subscription today!

Call Toll Free: DATATEL™ **800/341–1522** (For orders only, M–F 8 a.m.–9 p.m. Eastern Time)

Or mail this coupon to:
Sextant, Dept. S14
716 E Street, S.E.
Washington, DC 20003
Please allow 6–8 weeks for delivery of your first issue.

Your satisfaction with *Sextant* is completely guaranteed. If at any time you're not completely satisfied, just let us know and your money will be refunded—even for the issues you've already received.

Send me:

- ☐ **12 issues for $29.91 ($34.50 to Canada)**
- ☐ **6 issues (1 year) for $14.97 ($17.25 to Canada; $21.00 overseas via surface mail, $35.00 via air mail)**
- ☐ Payment enclosed (Checks must be in U.S. dollars payable on a U.S. bank.)
- ☐ Bill me
- ☐ Charge my: ☐ Visa ☐ MasterCard

Card # _____ Expires _____

Name _____

Address _____

_____

_____

Sextant, Dept. S14, 716 E Street S.E., Washington, DC 20003

work when running under 16-bit operating systems. If you have the Magnolia board, this means that you are stuck with your hard-sector drive(s) when running CP/M-86. As far as I can tell, MS-DOS won't run with hard-sector drives.

If you want to get full use out of your 16-bit operating system, you're going to have to dump your Magnolia controller and get something else. Heath and CDR controllers seem to work fine.

Incidentally, you need to tell TMSI which controller you plan to use when you order the H-1000.

You will also have some problems when you try to run many programs designed for the IBM Personal Computer. The H-1000 is billed as being IBM-PC compatible. To an extent, it is, since it will run PC-DOS. It will not, however, run most application programs designed to be used on the IBM. This is because a large portion of the current crop of IBM-related software ignores

Is the H-1000 for you?

Trying to decide if the H-1000 is a better deal for you than an equivalent H/Z100 depends a lot on *you.* The H-1000, for instance, has an edge over the '100 in its speed: 8 MHz instead of 5 MHz. But you have to decide if that difference in performance is enough to justify the greater cost needed to make it an equivalent to the '100.

You also have to decide if you are giving up too much when you get an H-1000. An inability to purchase software is a "cost," too. One of the very important items you give up is commercial software for the 16-bit side of the computer. There are a few commercial packages that can be run on the H-1000. But they are fairly rare, and are becoming more so. There are far too few H-1000s out there for commercial manufacturers to take notice.

To make matters worse, as the manufacturers of the programs converted from CP/M make changes, the changed programs are less likely to run on the H-1000. For example, dBASE II runs fine, as does Friday!. But Ashton-Tate's new dBASE III will not run on the H-1000. dBASE III will not run on the Z100 either, but Ashton-Tate is said to be making plans to release a Z100 version. An H-1000 version of this package is extremely unlikely. Only if TMSI is successful in emulating the IBM screen memory will you be able to access a substantial software base.

Total cost of ownership must also be considered. While the H-1000 sells for $1,495, there are other costs involved. Even if you don't wish to continue using your '89, it does have value itself, after all. Many Heathkit Electronic Centers will take '89's in trade on a '100 for a credit of about $500.

And if you don't have them, you must purchase a soft-sector disk controller and double-sided disk drives if you are to have a system equivalent to a '100. The cost of what you need can add up very quickly, especially if you have to buy any part of it. If you have a Magnolia controller, you should also consider that you will lose the use of it during 16-bit operation. And your memory-expansion board will be useless. Even if you can sell these items to help pay for a new controller, you will still lose something.

Let's take a look at the costs involved in buying a '100-series computer and an H-1000. (In making a personal judgement, I would base my calculations primarily on the kit version of the '100, because I feel that the audience involved would be more likely to buy a kit. The buyer of an H-1000 is more likely to be a kit builder.)

In this comparison, I will be looking at computers with similar memory, two drives, and a 16-bit operating system. Among the differences between the H-1000 and the '100 is the fact that a '100 will not currently run HDOS except by using an emulator. And it will not run programs written in Z80 code. Otherwise, both computers will run the same basic operating systems.

The comparative dollar figures appear in Table 1. They cover the H-1000 and the kit and assembled versions of the all-in-one H/Z120. The '120 configuration has two double-sided, double-density disk drives as standard equipment.

(Unfortunately, the figures can't be perfectly comparable. The '120 will hold only 192K on the motherboard. The H-1000 carries 256K. Although 64K S-100 memory boards are available commercially, they are not available from Heath. To get a value for the 64K, the cost of Heath's Z205-1 kit was used. In fact, though, the cost for an add-on board would be much more. The Z205 board with 256K costs $799. You run into similar costs, however, when you expand the H-1000 past its basic 256K. So it seemed better for us to stick with the lower cost, rather than get into endless comparisons of increasing RAM costs.)

By itself, the H-1000 is a lot cheaper than either the kit or assembled '100. But you have to have an '89 to make it work. You also have to have two soft-sector disk drives to make it equivalent to the '100s in Table 1. The most expensive alternative is the assembled Z120.

The most important comparison is between the H-1000 and the H120. As you can see from Table 1, the difference is about $800, with the H100 being cheaper. If you already have the drives and controller, your out-of-pocket cost is reduced; but your actual cost is still considerable.

The actual cost is even greater if part of your present system is rendered obsolete. Examples of this include memory-expansion boards and non-Heath accessories such as Magnolia Microsystems' soft-sector controller. You could sell them individually, of course. So you have to estimate what you might get for them. I suspect they add more to the value of the '89 than you would be able to get for them alone.

In making your comparisons, however, you should also note the fact that the H37 soft-sector system listed with the H-1000 will work in quad density with CP/M, CP/M-86, and HDOS. You should also note that all '89 software will work with the H-1000. Third-party manufacturers supply double-sided disk drives that can be configured to mount within the '89's cabinet—although this raises the problem of involving yet another manufacturer. Also, internal double-sided drives are subject to problems related to electrical noise, according to TMSI.

On the other hand, the '100 has superior graphics capability, can run

Z121-32 (assembled, all-in-one, 192K, two drives)	$3,729
Value of additional 64K*	100
	$3,829
H121-22 (kit, all-in-one, 128K, two drives)	$2,449
Z205-1 64K add-on	100
Value of additional 64K*	100
	$2,649
H-1000 (256K)	$1,495
Value of '89	500
Z89-37 controller	299
H/Z37-2 DS/DD drives	1,199
	$3,493

*See text.

Table 1. Comparative costs for the H-1000 board and kit and assembled versions of the H/Z120 computer. It should be noted that no value has been assigned to labor in constructing the H120 kit.

PC-DOS and works with the machine directly. (For more background on IBM compatibility, you might want to take a look at Graham Wideman's article in *Sextant* #9, March-April 1984, "Why Won't All That IBM-PC Software Run on My Z100?".)

Some of these IBM-PC programs address the screen memory directly to get faster response. Others use graphics, which makes them dependent upon the characteristics of the IBM video board. None of these programs will run on the H-1000.

A few programs do not go directly to the IBM hardware, and should run properly. Most of these programs are ones that were originally designed for CP/M, and then were translated to MS-DOS. Examples of these are dBASE II and Friday! by Ashton-Tate. Since these programs are also available for CP/M and CP/M-86, the ability to use PC-DOS does not really gain you as much as it would if the machine could run a wider selection of IBM-PC software. TMSI says that it is working on a solution to this problem.

You should also be aware of a few points relating to disk drives and the H-1000. If you have 80-track drives, PC-DOS will use only 40 of the tracks. Since the operating system was designed only for 40-track drives, that's all you get. In addition, there is some evidence that CP/M-86 has problems using 80-track drives reliably. However, TMSI says that this is not always a problem, and that a fix is in the works to take care of the situation when it does occur.

If you wish to work with both an H-1000 and an H/Z110 or '120, there are some compatibility restrictions that you should note. They involve version 1 of MS-DOS, such as the Zenith Disk Operating System (Z-DOS), and MS-DOS version 2, its upgrade. Versions 2 and over normally format disks with 9 sectors. This format cannot be read by earlier versions. If you work with a '110 or '120, you will have to tell PC-DOS 2.0 on your H-1000 to use the compatible 8-sector format. The later versions of MS-DOS can read either kind of disk. Therefore, your disks formatted on the '110 or '120 can always be read by the H-1000. So compatibility will be a problem only part of the time.

Documentation

The installation instructions for the H-1000 are excellent. As I said, if you built your '89, you should have no trouble installing and using the H-1000. In fact, if you have had even minimal experience inside your '89, you should be able to handle the swap. The troubleshooting section uses a flow chart that is exceptional.

Once you get past the installation, however, the level of the book changes. The reference section of the H-1000 manual is for experienced users and programmers only. As far as I can tell, no attempt was made to translate this area of the book into layman's terms.

Unfortunately, it is not a very readable manual. Printing was done using an offset method, reproducing a dot-matrix original. Margins are narrow, white space is at a minimum, and the overall appearance is quite dense. It would seem reasonable that an upgrade this expensive would justify a typeset, professionally done manual. Even if the volume does not justify typesetting, a letter-quality printer and more open page design would make the manual vastly easier to use.

Otherwise, the manual is adequate. The table of contents is complete. But there is no index; so finding the information you need can take a while. Almost half of the manual consists of data sheets for several of the ICs used in the H-1000, a useful addition for the advanced user.

The user who is less experienced will find little help in the reference section. Perhaps the H-1000 is unlikely to be purchased by an inexperienced user; but there are plenty of intermediate users who could use more extensive explanations.

The operating system software is accompanied by the manuals produced by their respective companies. TMSI has included addenda that explain the changes made by them, or that explain items unique to the H-1000.

Conclusions

For the most part, the TMSI H-1000 performs as promised. It allows you to have a computer that will run your 8-bit '89 software as is, and that will still have true 16-bit capability. You get a limited IBM-PC compatibility, the use of a lot more memory, and very fast operation.

All of this speed and capacity is expensive, however. The H-1000 is certainly cheaper than an assembled Z110 or '120. But the savings are eliminated if you're willing to go to the kit version, the H110 or '120. And depending on the hardware you already have, you might find that you have to spend more with the H-1000 to get a system equivalent to the '100. (Accompanying this article are some comments regarding the costs of getting started with each.)

If you need IBM-PC compatibility, the kit version of Heath/Zenith's '150 computer is available. For $1,595, you get a single-drive version which is essentially 100% compatible.

On the other hand, if your needs dictate the requirement for fast operation, the H-1000 will run very well. TMSI provides very good support for their product, so you should have few problems installing and using the H-1000.

Ordering Information

H-1000U, $1,495.
Technical Micro Systems, Inc.
P.O. Box 7227, Dept. A
366 Cloverdale
Ann Arbor, MI 48107
313/994-0784

color graphics with the addition of color memory chips, has the industry-standard S-100 bus, and runs 8" disk drives without modification. Since the H-1000 is a converted '89, only the standard graphics package is available unless an add-on graphics board is purchased at additional cost.

In performance, of course, the difference is that the H-1000 runs at 8 MHz; the '100 at 5 MHz. Is the difference in performance worth $800? That depends on your needs. Remember that the difference is only in the 16-bit mode. When running 8-bit programs, the '100 is faster, running at 5 MHz, rather than the H-1000's 2 or 4 MHz. If your operations are very CPU intensive, the speed of the H-1000 will probably be of interest to you. If you do not already own an '89 with the Heath or CDR soft-sector controllers and soft-sector disk drives, and if commercial software is important to you, the '100 seems to be the logical choice.

"The logical choice," however, depends on your own circumstances. What if you don't own an '89? The H-1000 board also can turn an H/Z19 terminal into a full computer. The trade-in value of a '19 is around $300. So if you have a '19, knock off a couple of hundred dollars or so from the cost of a total H-1000 system. (Of course, you still need a disk-controller board and drives. And you would need a power-supply upgrade, whose parts would cost about $100.)

Finally, what's the relative value of your labor? I based a lot of my comments above on the assumption that building a Heathkit H100 was the realistic alternative to TMSI's H-1000. What if you don't have much time to spare? An H100 kit will take 40 hours or so, assuming all goes well. (See "Building Heathkit's H120," by Hurley D. Cook, in *Sextant* #7, Fall 1983.) The H-1000 is a much simpler job and can be handled in an afternoon.

So, no matter what your needs and resources, quick comparisons can be misleading.

Oh well, kitbuilders are fond of saying that "getting there is half the fun." You just have to remember that "getting there" includes deciding where you want to go.

10 MB WINCHESTER DRIVE

LINDLEY SYSTEMS ANNOUNCES MegaDisk, a 10 Megabyte external

Winchester hard disk drive for Zenith Z-100 and Z-100PC (Z-150/160) computers.

The package includes all you need to get up and running quickly - the Winchester drive, its cabinet and power supply, the cable, the controller card, and the software for MS-DOS (tm) (Versions 2.0 or higher).

Shipment via UPS to Continental US locations. Certified check will speed your order. Shipping & insurance included in prices.

For the Z-110 and Z-120 computers: (order No. 801)

SYSTEM PRICE: $ 1299.95

For the Z-150 and Z-160 computers: (order No. 802)

SYSTEM PRICE: $ 999.95

External system for use with Zenith controller and software: Call for details.

The above ad shows some of the numerous print sizes and densities available with our UPC drivers (see below for more information). Note the double height and double-height double-width printing. From 16 to 64 different sized characters are available, depending on your printer.

M A I L B A G

* Generates lables and form letters.
* Sorts by name, ZIP code, other fields.
* Set label size and number across.
* Any number of names (up to disk size).

Order for H-8/89/90 HDOS or Z-100 Z-DOS. Requires Microsoft Basic. $25.00 ppd.

U P C Graphics Printer Driver

* Prints H/Z-89 screen graphics.
* Prints directly from program or file.
* Prints "picture" files saved from Software Toolworks Ed-A-Sketch.
* Easy to use character font generation program included.

Specify printer and system. $25.00 ppd.

U L T I M A T E Printer Driver

* Control printer from keyboard or file. Tilde (or selected character) makes next character a control character.
* CP/M version has on line SET feature.
* HDOS version has CTRL-P screen print.
* Special codes for overstriking and making long lines.
* Both parallel and serial interfaces are supported.

Specify printer and system. $20.00 ppd.

Above are for CItoh 8510, Epson, Star, MPI, NEC 8023, Microline 82/83/92/93 H/Z-8/89/90 HDOS or CP/M-80 Systems

Z-UTIL Utility for Z-100.

SCREEN - Dot graphics screen dump for Epson, Star, CItoh 8510, MPI, NEC 8023, Tally, and Microline.

TP - Single screen editor - great for notes and printer setup.

RUNPC - IBM-PC Emulator which allows many programs to run on the Z-100. $35.00 ppd

CHECK MASTER

* Automatic checkbook balancing
* Archival file for tax records
* Easy-to-use menus for changing, deleting, and adding new checks
* Automatic Statement processing
* Specify HDOS, CP/M, Z-DOS, PC-DOS, MS-DOS. $20.00 ppd.

DIAMOND ACCOUNTING INVENTORY

* Automatic links with A/P (auto. back order), A/R (deplete your inventory as orders come in).
* Changeable fields; menu-based.
* Specify HDOS or ZDOS. $35 ppd.

LINDLEY SYSTEMS 21 Hancock Street
Bedford, MA 01730 (617) 275-6821
Write for FREE product catalog!

Circle #132 on Reader Service Card

WatchWord: Custom Built for You and Your Z100

Using its macro capability, you can extend and change WatchWord to meet your needs.

Beaufort Lancaster

Have you ever wished for the word-processing power available in the $300-and-up price category, but just couldn't see the justification for tapping an already strained bank balance any further? (Or couldn't get the boss to see it?)

Well, what would you say if I told you there was a word-processing program which not only has most of the features of the famous name brands, but also adds a few features that most do not have? It runs on the H/Z110 or '120 under the Zenith Disk Operating System (Z-DOS).

But I hear you asking, "Get to the point. Just exactly how bad is the bad news? What's the price?"

Well, a lot of you already know the answer. The product is WatchWord; its author is Steven Robbins; and it comes from S & K Technology, Inc., of San Antonio, Texas. The price is only a third of the $300 you might expect.

If you have an H8 or an H/Z89, you may already know about Steve and S & K from using WatchWord's predecessor EDIT19. It's still available for only $80. And its partisans tell me it makes its well advertised competitors "look like stone axes" by comparison.

But you don't have to be satisfied with what you hear others say about an earlier product nor with what you read here about WatchWord. For the paltry sum of $3 (which includes shipping and handling), you can get your own copy of a demo disk of WatchWord. You can try everything described here till you reach a saturation point. Except for the printing features, that is. Savvy programmer that he is, Steve figured he'd better leave some essential feature disabled on the demo disk or he'd wind up with little financial reward for his considerable efforts.

At $3, you shouldn't be surprised to learn that the demo disk doesn't come with a printed manual; but all the instruction you need is contained in .DOC files on the demo disk itself. In fact, the demo disk contains essentially everything that is in the printed manual that comes with the full program.

What it will give you

WatchWord is about as close to "what you see is what you get" as it possibly can be. It includes on-screen underlining, boldface, strikeout, superscripts, and subscripts. These five specialized text features are activated in WatchWord by control sequences.

Other features include centering, formatting, microjustification (on specialty printers), autowrap, autosave, split screen, macros, user-settable defaults, and user-definable keys. Lines longer than 80 characters are permitted, with automatic horizontal scrolling as the cursor moves. The colors for display screens, separator lines, and messages are user-selectable.

All of these delete keys become "undelete" keys when they are preceded by the ESC key.

As for speed of operation, the program is relatively fast in executing commands because it's written in 8088 assembly language. It also doesn't hurt that the entire program is in memory at one time. There are no delays caused by the program's stopping to read in an "overlay" from the disk. The program is fairly large at about 85 kilobytes, not including HELP and DOC files.

Lest the preceding list of features scare away the beginner, let me hasten to add that although the program is very flexible and powerful, it is still easy to use. You can use the program right away without having to master all of its more sophisticated capabilities. But they are there when you are ready to delve into them.

Maximum use is made of the Z100's special-function and numeric-keypad keys, shifted and unshifted. You can execute the most frequently used functions with one or two keystrokes. The Z100 HELP key is used for exactly that—pressing it at any time brings up a help-screen menu of items to select for explanation. These are generally condensed reminders of the detailed discussion in the manual, not elaborate examples or extensions of what is said in the manual.

Documentation and support policies

As for the documentation, I should mention that it is very attractively printed using a letter-quality printer. But it is understandably not of the caliber of manuals accompanying programs which cost three to five times as much. The manual does seem to have been proofread and edited fairly carefully so that it is free of typographical or substantive errors.

The manual for WatchWord version 1 was not indexed. But the manual's design was such that I did not really miss an index. The new manual keeps the same format. The style is tutorial, with 19 lessons of one to eight pages each. It has a thorough table of contents. There is also an alphabetical list of commands at the end of the manual and a list of function keys by keyboard location. It has grown from 57 to 80 pages, with the index.

(The documentation is further discussed by John Walker, *Sextant*'s technical editor, in comments accompanying this article. He also discusses WatchWord's macro capability, which allows a complex series of commands to be stored in a disk file.)

An identifying strip is provided to be placed above the Z100's special-function keys, labelling their functions; but it is merely printed on a long, narrow strip of thin paper. I chose to sandwich mine between two pieces of adhesive-coated plastic laminating material such as that used for covering drivers' licenses, etc. This stuff can usually be found at the office supply store of your choice.

As you are probably beginning to see, this program includes a lot of power and functionality for the money; it does not provide stick-on key caps or slick, glossy manuals with multi-color illustrations.

In keeping with the practice of many independent Heath/Zenith vendors, S & K has not copy-protected the disk.

There is little evidence of the thoughtlessness exhibited by some other vendors in their backup disk policy. As for more direct customer support, on my one occasion to telephone with a question, I left a message on their answering machine. My call was returned later the same day and was dealt with in a courteous and technically competent manner.

S & K Technology states that future updates will be either free or at nominal fees. That will depend upon whether the reason for the update is to correct a legitimate bug, to offer additional features, or a combination of these. For example, upgrading to version 2 costs $10 if you bought after April 1, 1984; $15 if you bought before that.

Getting started

To begin using WatchWord, you simply call it up from the Z-DOS prompt by typing the drive specification and WW. It will come up with the cursor in the text area of the screen; and you are ready to type away. A file name can be assigned to your text at any time by hitting ENTER to move the cursor to the command line, and typing in the file name. To bring in an existing file named FILENAME.DOC from a disk in drive A, type the command

READ A: FILENAME.DOC

on the command line. From Z-DOS, typing WW and the filename after the A: prompt will start WatchWord with the desired file in memory.

Most commands which initiate a disk read will search other drives if the file cannot be found on the named drive. You may change disks in any drive at any time—unless you are using the feature that allows editing a file which is larger than memory.

If the AUTOSAVE n command is in effect, the file is automatically saved to the disk after every n changes as you edit. The n is user selectable. If you tend to get too carried away with the flow of your thoughts to type SAVE occasionally, the ability to have the program automatically stash your creation away to disk is very nice. A counter on the status line shows the number of changes remaining before the next AUTOSAVE will occur.

Another nicety is the DIR command for displaying the directory of files on the disk. It works the same under WatchWord as DIR/P does under Z-DOS. This means you can get the directory listing without having to exit the program. If the directory is too large to fit onto the screen, the program very politely stops after filling the screen and prompts you to press any key to continue with the directory listing.

Screen organization

The default screen configuration has a text area of twenty lines at the top of the screen. Below this is a zig-zag line separating the text from a two-line area that is used for error messages and command entry. Then comes a solid separator line followed by a status line. Pressing the ENTER key alternately toggles the cursor between the text-entry area and the command line. The status line continuously shows a number of things: the current cursor position by line and column number; the current file name and size; the amount of free memory space available; a time-of-day clock; and the status of the various modes of text entry (whether you are in INSERT mode, etc.).

If this screen configuration is not to your liking, however, do not fret. A simple-to-use, menu-driven configuration program gives you the option of changing the screen format—as well as just about every other characteristic of WatchWord, including function-key commands. Let's face it, things just don't get much more flexible than this.

Moving from here to there

Cursor movement is accomplished by the arrow keys in the expected manner. One- or two-keystroke commands will shift the cursor to the beginning of a line, end of a line, top of a text file, bottom of a text file, and forward or backward by word or paragraph. When the cursor reaches the sixth line from the top or bottom of the page, pressing the arrow keys causes the text to scroll up or down one line at a time (without having the entire page repainted).

There are also commands for scrolling the text up or down one page at a time. There are no commands to move the cursor immediately to the top or bottom of the current screen display or to step the cursor by sentence. The keystroke commands are supplemented by the commands DOWN and UP, which may be followed by a number to indicate the number of lines of desired movement.

Handling text

For moving text, special-function keys are used. They can split a single line into two lines at the location of the cursor, or join the line containing the cursor with the line below. And they are used for cut, copy, and paste operations.

Text to be copied or moved may start at any point within a word or line, not just at the beginning of a word or line. For text entry, the default mode is to overwrite or replace existing text. The insert-character mode can be toggled on and off by using the ICHR key at any time. In the insert mode, the letters IC appear on the status line.

Text deletion using the Z100's DELETE or BACKSPACE keys moves text and the cursor to the left and deletes characters to the left of the cursor. Using the DCHR key keeps the cursor stationary and deletes the character at the cursor position while shifting left the text that is to the right of the cursor.

An entire line may be deleted using the DEL LINE key. The DEL n command or -DEL n command will delete n lines below or above the current cursor position.

All of these delete keys become "undelete" keys when they are preceded by the ESC key. The ESC key acts as a switch and remains in effect until any key other than a delete key is pressed. Up to 100 individual characters or 20 lines may be undeleted in this manner. Additionally, if an error is made in editing a line, a single keystroke will restore the line to its original form so long as the cursor is still on the line to be restored.

WatchWord also has a split-screen function which is useful for editing two files simultaneously, two parts of the same file, or two different versions of the same sentence or paragraph. To help keep things straight, you may find it useful to set different colors for the two display screens. This is easily done on the command line or from the configuration menu.

Text may be moved or copied from one window to the other using the paste buffer with the cut or copy keys. To combine two files, the command GET FILENAME will insert an existing file, FILENAME, into the current file being edited at the current cursor position.

LOCATE and CHANGE commands

A global locate, or "find," command works either backward or forward from the present cursor position. You may choose either to ignore or not to ignore the difference between upper- and lowercase during searches.

WatchWord is about as close to "what you see is what you get" as it possibly can be.

The CHANGE (search and replace) command searches only from the present cursor position to the end of the file. However, if you want to do a search and replace on the entire file, it only requires a single (shifted) special-function key to position the cursor at the beginning of the file.

The CHANGE command can also be restricted so that it affects only user-selectable zones or columns. At your choice, the CHANGE command will change all occurrences of a string; or it will stop at each occurrence and allow you to approve or veto the change before going on to the next occurrence. You may also elect to cancel the command at any point without continuing to the end of the file.

Other key commands

Other special-function-key commands (some require the shift key also) include a change-case function, delete to the end of the line, switch split screens, redisplay error message, redisplay the screen, and change the high bit of a character. When entered at the beginning of the command line, the "=" key will repeat the most recent command.

WatchWord's normal configuration uses the numeric keypad for a number of functions. The functions include setting and clearing tab stops, forward and backward cursor movement to tab stops, and decrementing or incrementing the left margin. If you wish to use the key pad to enter numbers, however, the command MAP KEYPAD allows this.

Formatting

Under WatchWord, the FORMAT command simply moves the text to fit within your margins. A long word that would extend past the right margin, then, shows up on the next line. This may leave the first line noticeably short. If you want to hyphenate the word, you have to do so by hand. (But if you later reformat, you'll have to remove the hyphen and space if they show up in the middle of a line. WatchWord doesn't presently support "soft" hyphens that disappear when not needed.)

The usual left-margin, right-margin, and right-justify commands are available. Paragraphs may be formatted with indentation of the first line by giving the number of spaces to indent in an INDENT n command followed by an IFORMAT command.

The command FORMAT will format a single paragraph. FORMAT n will format n

lines. To change the implied formatting from n lines to n paragraphs, the command PARA FORMAT should be entered. This paragraph reference remains in effect until the default mode is reset by entering the command LINE FORMAT.

The command HFORMAT will format paragraphs with hanging indentation. This is used to permit numbered paragraphs to keep the numbers aligned in the left margin while formatting the rest of the paragraph as usual. A ZFORMAT command will format only in the columns set with the ZONE n1 n2 command, where n1 and n2 are left and right column numbers.

Printing

WatchWord implements underlining, super- and subscripts, boldface, and strikeouts via combinations of the CTRL key and various single letters. Special fonts are used to assign unique symbols to mark these functions in the text on the screen. These may be displayed exactly as they will print by using the ENHANCED or SUPER ENHANCED viewing modes (except for boldface, which is shown as reversed video). When the text is sent to a printer, these control sequences are converted into combinations of backspaces with or without forward or reverse line feeds, as required.

Many dot-matrix printers, such as my Okidata 92, do not accept these commands as such. However, the command NO BACKSPACE allows the Oki to produce strikeout and boldface correctly.

The configuration program makes using supported printers very simple. WatchWord may be configured for these printers either by the configuration menu program or by typing your printer's name on the command line. The printers supported are the Diablo, NEC Spinwriter, Epson, Gemini, H/Z25, M.P.I., Okidata (82, 83, 92, and 93), and Prowriter printers.

Even if your printer is not supported by WatchWord's configuration program, however, it's not that hard to get them talking to one another. When I was working with version 1 of WatchWord, my Okidata 92 was not yet supported. But using the instructions and examples in the .DOC files, I was able to assign the ESCape sequences required by the Okidata to the special screen-display modes. These ESC sequences had to be assigned to different CTRL combinations from those used by WatchWord for specialty printers, however. (Although it was not entirely necessary to do so, I also assigned the special screen symbols displayed by the WatchWord CTRL combinations to my CTRL combinations.)

To enable the other print capabilities of the Oki, I used the CREATE command to invent names for the functions and assigned the required ESC codes to these names. These commands are recognized as dot commands (discussed below) and activate the various printer functions when they are inserted in the text.

I could also have accomplished the same ends simply by embedding my printer's control codes in my text wherever they were necessary. WatchWord makes this easy with its special-character insert key. With this, you input hexadecimal codes directly into the text; hex 1B for the ESCape character, for instance. (You may discover that finding your printer's control codes is the hardest part of the job.)

For a number of printing capabilities, WatchWord, like many other word processors, employs dot commands—instructions preceded by a "." in the text. These are interpreted as printer-control commands and are not printed as part of the text.

Among the features controllable by WatchWord dot commands are:
10 or 12 characters per inch;
6 or 8 lines per inch;
page headers and footers;
pause or no pause between pages;
blank lines or no blank lines (to skip perforations) between pages;
double or single spacing;
start or end page;
microjustification on or off;
set number of lines per physical page;
set current page number to n; and
start new page unless at least n lines remain on this page.

Some of these may also be entered on the command line, as well as in the text.

WatchWord at *Sextant*

At *Sextant*, we tend to be partisans of WatchWord and its 8-bit predecessor, EDIT19. And with good reason—without them, our editing would be a lot harder, and our typesetting bills a lot higher. Over the last few years, we've sent several pages of comments, criticisms, and dumb questions to Steve Robbins, the author of WatchWord and EDIT19. (I harass him at conferences, as well.)

Some background may help. When an article is accepted, the author sends in a copy on disk. We do all our editing at the keyboard. And besides editing the article, we put in all the typesetting codes that indicate type size, italics, boldface, etc., etc., and a lot more etc. Then we send the disk to our computer typesetter (Micro World Publishing of Northfield, Minnesota). It saves money since they don't have to re-type everything. And it saves correction time since most mistakes in the typeset copy will be ours. EDIT19 and WatchWord have both proven ridiculously easy to use to do both the editing and the typesetting work.

One reason I say "ridiculously easy" is that some Big Time Computer Magazines still take time to warn their authors about the supreme necessity for double-spaced copy and the wickedness of using erasable paper. I can only assume that they're still editing articles on paper—with blue pencils and eye shades, and rushing the marked-up copies to folks sitting at steam-powered Linotype machines. That may be great for a flick about newspapers before the turn of the century, but this is the wrong century.

Most of us here at *Sextant* came to the job without much (or any) computer background. Both EDIT19 and WatchWord have proven easy to learn how to use. They're designed for the keyboards of the H/Z19 terminal and the '100 computer. The commands needed to accomplish most jobs are easy to remember. I've seen some of the menu-driven word processors. And I'm convinced they'd drive me nuts—at three times what Steve charges to do the job conveniently. If I want to change the first three x's found on each of the next five lines to y's, I find C/x/y/5 3 a lot easier than wading through some long menu. Most of Steve's commands are quite straightforward.

But a lot of word processors will

as dot commands.

Page breaks are not shown on screen. However, the commands FIND PAGE n and FIND NEXT PAGE will locate the page breaks without requiring you to print the file.

There is a fair amount of flexibility in the print command. Printing may be interrupted at any point by a CTRL-C and continued by PRINT *.

The print options available include PRINT n (to print n lines beginning at the current line) and PRINT PAGE n (to print the single page numbered n). The output may be sent to the disk rather than the printer by the commands FPRINT FILENAME n and FPRINT PAGE FILENAME n.

FPRINT is useful if you wish to save a file and print it later directly from Z-DOS. Saving it under FPRINT, you will still retain all the print-formatting features you would have if you printed it from WatchWord.

FPRINT is also useful in debugging the control sequences needed for printer configuration—since it saves the file exactly as WatchWord would send it to your printer. For instance, words in "software boldface" (CTRL-X) will show up as carriage returns (without line feeds) followed by the number of spaces needed to line up the words, and then the words themselves for re-typing. If you had an H/Z25 printer, and put in a CTRL-Q for a "hardware boldface," you would find the CTRL-Q's had disappeared from the file saved with FPRINT—the '25 does not have a boldface feature built in, so WatchWord simply drops the CTRL-Q.

FPRINT is just one example of WatchWord's flexibility. You can use it as just a convenient printing tool—or you can use it to "go inside" when you want to adapt WatchWord to more complex uses.

Some shortcomings

Of course, no program has everything. This one has a lot, but there are some things missing which may or may not be important to you personally. Page headers and footers are limited to a single line. There is no automatic procedure to help you put in whatever hyphenation you wish. I mentioned the cursor movement limitations earlier.

There is also no mention or discussion of error messages in the manual. In fairness, though, I should add that these messages are pretty much in plain English—rather than programmer's jargon or numerical codes. The context in which the errors occur generally is sufficiently clear so that no further explanation is needed.

As noted above, S & K offers an update free or at nominal charge to anyone who requests a bug corrected in any of the supported features described in the manual. So far, I haven't really needed to claim a free or nominal-charge update, however.

One bug did show up in version 1: the FORMAT command would work properly for only one paragraph at a time. Trying to format a number of paragraphs at once would get you a "word longer than line" error message. Before I got back to Steve on that, I was already working on this review with a preliminary copy of version 2—in which it was corrected.

Two bugs did turn up in the preliminary copy of version 2. These were in features which worked perfectly in the earlier version: the zone format (ZFORMAT) and the dot command for double spacing (.Double). These bugs were reported and have been corrected.

Altogether, then, even WatchWord's few bugs point to its value. They've been *fixed*—quickly and without the bureaucratic sidestepping that shows up in some software companies. The product, its support, and its low price make as powerful a combination as microcomputer users are likely to find anywhere—and it's only available for the Z100.

Ordering Information

WatchWord Version 2, Z-DOS, $100.
Demonstration disk, $3.
EDIT19 Version 3.1, HDOS only, $80 (with reference manual) or $95 (with reference manual, tutorial manual, and tutorial disk).
S & K Technology, Inc.
Attn: Steve Robbins
4610 Spotted Oak Woods
San Antonio, TX 78249
512/492-3384

search and replace for you. Steve's virtue is flexibility. Most word processors are written to do the things their authors expected you'd want. He seems to have written his programs for folks who want to do things he might *not* have thought of. For us, the macro capability has proven indispensible.

Macros

Editing, after all, is just editing. We may delete more or insert more or move things around more at a magazine than you might do on a report you were writing. But the job is basically the same.

When it comes to putting in all the typesetting codes, however, we have a potential headache that most people don't have to worry about. If you look through an article, you'll see we change styles and sizes frequently. We use a slightly smaller typesize for capitalized "computer" terms like GOTO or EDIT19. We use a slightly larger size (in boldface italics) for the subheads at the beginning of each section. Going back and forth between all of these, we can rack up as many as a thousand changes in a long article. A hundred is minimal. And the typesetting codes for each of those changes are at least four characters long. (\ FT2 takes us to typefont 2, italics, for instance. \ FT1 takes us back to our regular roman typeface.) The code for a size change is six characters long.

Could you imagine typing in that garbage every time it was needed? We make the job easier by using simple symbols in place of the typesetting codes. { and }, for instance, begin and end italics. << and >> are used for subheads. But that still leaves us with the job of translating those to the full codes. Even if it were just a matter of changing every { to \ FT2, it would be a nuisance. And things get complicated—what about when we want to use curly brackets and backslashes in the article? (As here.)

And what about proofing? We don't want to go into italics and forget to come out.

With macros, the job is a lot easier. A macro is simply a series of commands kept in a disk file rather than being typed in at the keyboard. We made up a series of macros to do the dirty work for us. They do all the changing. They were written for EDIT19 when we started out, then re-written a bit (and souped up a little) to run under WatchWord.

The macros also do a lot of our proofing. They count the number of {'s and }'s, for example. If they don't come out even, we know we've got an error. And another macro helps us look for it. (True, we could have made two offsetting errors, but it doesn't usually happen. Since we started using the macros, we've had only about three or four typesetting errors they missed. If there's an error, it was put in *after* the proofing macros were run.)

Under WatchWord, the macros can be more than just a simple list of ordinary commands. For instance, you also have the ability to store values in memory. And you can set up subroutines to access them as needed. You have, in effect, a mini programming language.

How useful might the macro capability be to you? If you do anything to one file that you do to another, a macro will save you typing. It could be as simple as reformatting documents. Or it could be something as tedious and error prone as putting in typesetting codes. The macros are at

least a time saver and possibly a life saver.

Flexibility

But other word processors have a macro capability (albeit at three times the cost of EDIT19 or WatchWord). What they usually don't have is WatchWord's flexibility. Not even EDIT19 is as flexible. For instance, I don't care for the keys labelled on the '100 for cursor control. So I've written a macro to change the numeric keypad to emulate the '19's. If I were masochistic, I could also control the cursor with the control characters employed by WordStar. (Up arrow is CTRL-E, etc.)

Some changes can be incorporated into WatchWord by means of its configuration program. Others are done with macros. Still others can be done either way. Practically everything that happens in WatchWord is changeable. You can disable functions; and you can execute functions with a different command-line entry or special-function key than Steve has set up.

This flexibility is more than just fun. If you've gotten used to EDIT19 or some other word processor, it can avoid a lot of frustration. If needs be, you could configure the keyboard and command language to the taste of *each user*. If you perform some functions a lot, set up WatchWord to do them with single keystrokes. Write up a macro to set things up automatically. Once you learn how, you'll be surprised at how easy it is.

Documentation

But finding out how to do the job is too often a lot harder than actually doing it. In some respects the documentation is outstanding; but, particularly for the beginner, it can be tough sledding. In fact, if I had to list all of WatchWord's significant weaknesses, documentation would head the list. It would *be* the list—because practically anything else I might not like is easily overcome.

The documentation, though, has sent me up the wall more times than I care to count. The WatchWord manual is written as a tutorial. I don't want and I don't have the time for a class. I want to go to an index and find how to do a particular job. Instead, I have to go back and forth and look around a number of places where I suspect Steve has hidden the information I want. It'll be there—with just that occasional dash of obscurity to make me try the job three ways wrong to find out the way to do it right.

I don't want this to sound too negative. And if I *had* the time to use the manual as a tutorial, I'd get a lot more out of it. The tutorial/manual seems to be a popular idea. But I'm sorry—manuals are a fetish of mine. Most of the time, they're all I have to tell me what's going on. And when you're in that position a lot, you can learn to *hate* manual writers.

But with Steve, there's a difference. I would weep as I made out his execution order. What else can you do with someone who will placidly bury something like this in his documentation:

"The user may not be able to figure out the action of these [flags] from the brief descriptions given. The information is included here for those who would like to consider WatchWord an 'Adventure' type game. If more information is requested by enough users, more will be provided in future releases."

But—like a good Adventure game—it's all there. In some respects, in fact, WatchWord's documentation is perhaps the *best* available for any piece of software.

If there's anything happening inside WatchWord, he's told you about it *somewhere*. Besides the 86-page user's manual, there are nearly 120 kilobytes of .DOC files that go along with it. If you want to do anything tricky with WatchWord, the first thing you should do is print those files to paper. (Do so under WatchWord—they'll be formatted and numbered for you.) Whatever you want is probably there.

If you're a novice (or helping one), you may not want to try anything tricky. In that case, everything you want is probably in WatchWord's HELP files. You might print them out as a mini-manual. If you have the time to go through the tutorial, I think it will be worth it.

If you want a great inexpensive word processor for use in an office environment where you've got a variety of users, you should at least buy a demonstration copy ($3) to try out. The price alone should indicate that. But if you want to go further, you should have someone in the office who likes Adventure games. There's value in WatchWord's tunnels that will be worth the cost of exploration.

Anything else?

As you may have guessed, I tend to be more fanatical about manuals than other folks. So you can take my comments with a grain of salt. But have we actually found anything *wrong* with WatchWord? Well, a little.

It would be nice for us if WatchWord supported true proportional spacing. That is where you can assign different widths to different characters. (An "m" would be three times as wide as an "i", say.)

Proportional spacing, of course, is most important if you want right justification. WatchWord does, however, support microjustification and "fancy microjustification." (In right-justified copy with many word processors, extra whole spaces are inserted between words to make the right margin line up. This can look uneven. Microjustification throws in fractions of spaces; and "fancy" microjustification also moves the characters in a word closer or farther apart to get a more even appearance. But even fancy microjustification still won't give right-justified copy the polish of true proportional spacing. And most authorities recommend against it.)

Other than that, EDIT19 has a nice way of handling lines longer than the terminal width. The last character on the right is in reverse video. In WatchWord, there's no warning. (But it's a *lot* easier than in EDIT19 to go out there and see what's there.)

And in EDIT19, you can quickly see whether you're in command mode or in text-entry (edit) mode. In command mode, the whole current line is in reverse video. In WatchWord, there's only an unblinking block cursor, while the normal cursor moves to the command line.

Finally, when you go to the "bottom" with WatchWord's special-function key, you actually go to the first character of the last line, rather than the end of file. And sometimes its response seems to be a tad slower than I'd like. And when I type in a bad command line, the error message wipes out the bad command, so I can't see what my blunder really was.

Well, I've looked for things to criticize and that's what I came up with. Personally, my response to WatchWord is very simple: its only competitor is EDIT19.

I do some writing on the side and I need to get a computer to use at home for word processing. I've tried other word processors here at the office, read favorable reviews of still others, and talked with people who use other products. Sometimes I've found it difficult to believe that so many people can settle for so little and pay so much to get it. Even if I didn't care for the equipment (I do), I'd still try to buy an H/Z110 or '120 computer—just to make sure I could get the word-processing power available in WatchWord.

If a '100 is too expensive, the only other choice is an H8 or H/Z89 with EDIT19.

—John Walker, Technical Editor

RESOLUTION, SPEED, AND SILENCE

Superior Graphics and A Pseudo Disk for $495.

That's right, silence! Because when you plug the Interactive Graphics Controller/ Pseudo Disk into your H/Z89 or H/Z19-H8 system you not only get graphics that rival most 16 bit computers, you also get a high speed Pseudo Disk that does the work of another disk drive in your system, only much faster, and without all of that noise!

This upgrade stretches the horizons of your system into the future with High Resolution Graphics, Pseudo Disk Storage, 2 Trackball/ Joystick Interfaces, and 2 Parallel Printer Ports, all for one very competitive price.

Copy your CP/M or HDOS files to the Pseudo Disk and cut execution times as much as 80% over mechanical drives with disk I/O intensive programs. Files are even retained when you cold boot, and HDOS users can sysgen it as the boot device!

Take a good look at these features: • 3 pages of graphics, 640 by 250 pixels each • Mix text and graphics with independent scrolling • A remarkably easy-to-use high level driver that gives ANY language the power of fast graphics • Unmatched drawing speeds result from an advanced design which requires no handshaking or wait states • Use interlace mode and double the screen resolution to 640 by 500 pixels (requires Super19 or Ultra ROM) • Plug in up to 2 trackballs or joysticks for interactive applications (provisions for light pen) • 2 Centronics parallel printer ports with full support available • An alternate character set ROM socket to double the number of characters your screen can display (Script character set available) • Source codes available • European 50Hz compatible • Mounts professionally and easily into your H/Z89 or H/Z19-H8 and no existing features are sacrificed • 90 day limited warranty.

COMPARE	Northwest Digital Graphics-Plus	Cleveland Codonics Imaginator	SigmaSoft and Systems
Normal Pixel Resolution	512 × 250	504 × 247	640 × 250
Interlaced Pixel Resolution	n/a	n/a	640 × 500
Graphics Memory Size	16k	16k	64k/128k*
Number of Graphics Pages	1	1	3
Pseudo Disk Memory Size	n/a	n/a	64k/128k*/256k*
Super19/Ultra ROM Compatible	No	No	Yes
Vertical Graphics Scrolling	No	No	Yes
Type of Interface	Serial	Serial	Parallel
Independent Power Supply	No	No	Yes
Graphics Input Device Ports	n/a	n/a	2
Parallel Printer Ports	1*	n/a	2
Price/Performance Ratio	Good	Better	Best

*Option at additional charge.

Call or write NOW so we can rush our complete information package to you!

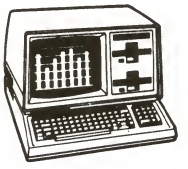
SigmaSoft AND SYSTEMS

"Support is the most important feature."

4488 Spring Valley #107, Dallas, TX 75234, (214) 392-1025

Circle #185 on Reader Service Card

HEATH/ZENITH SPOKEN HERE . . .



ZENITH IBM COMPATIBLE COMPUTERS

ZF-151-21 Computer. One 360K disk drive, 128K RAM, IBM bus with 4 open slots, color ready. Monitor sold separately. Retail $2199.00, Payload $1752.00
ZF-151-52 Computer. As above with two disk drives and 320K RAM. Retail $2799.00, Payload $2147.00
ZW-151-52 Computer. As above but with 10.6 MEG Winchester Drive and one 360K disk drive. Retail $4799.00, Payload $3320.00
ZFA-161-21 Portable Computer. 9" Amber screen, About 33 lbs. All other features same as ZF-151-21. Retail $2399.00, Payload $1883.00
ZFA-161-52 Portable Computer. 9" Amber screen, Two 360K disk drives. All other features same as ZF-151-52. Retail $2999.00, Payload $2280.00

ZENITH LOW PROFILE COMPUTERS

ZF-101-31 Computer. Single 320K disk drive, 192K RAM. Monitor sold separately. Color RAM set required for color. Retail $3029.00, Payload $2479.00
ZF-111-22 Computer. Two 320K disk drives, Color Graphics, 128K RAM. Monitor sold separately. Retail $3499.00, Payload $2807.00
ZW-111-32 Computer. Internal 11 MEG Winchester hard disk drive, One 320K disk drive, Color Graphics, 192K RAM. Monitor sold separately. Retail $5499.00, Payload . . . $4172.00

ZENITH ALL-IN-ONE COMPUTERS

ZF-121-32 Computer. Two 320K disk drives, 192K RAM, Monitor built in. Retail $3729.00, Payload $2905.00
ZW-121-32 Computers. Internal 11 MEG Winchester hard disk drive, One 320K disk drive, 192K RAM, Monitor built in. Retail $5599.00, Payload $4228.00

ZENITH MONITORS

ZVM-135 Color 13" High resolution, RGB & Composite . . . $493.00
ZVM-133 Color 13" High resolution, RGB only $447.00
ZVM-122 Amber 12". . . List $170.00, Payload price. . . $119.00
ZVM-123 Green 12". . . List $150.00, Payload price. . . $108.00
ZVM-123A Green Non-Glare 12". . . Payload price. . . $119.00
Cable for color monitor. Specify Computer ..... $ 25.00
Z-29 SMART DATA TERMINAL Detached keyboard, 12" green $623.00

MSDOS SOFTWARE FOR ZENITH COMPATIBLES

We have close out prices on several software packages. Please write or call for our up to date listing.

SOFTWARE CLOSE OUT FOR ZENITH H/Z-89

Last call. Only about ten remaining out of the original lot of 300. HSC-817-1 SuperCalc for CP/M, Hard Sector $60.00

UPGRADE ACCESSORIES FOR H/Z-89 COMPUTERS

Magnolia Microsystems Double Density Controller. Soft-sectored disk controller. Supports up to four each 5.25" and 8" disk drives. Complete with cables, installation instructions and CP/M 2.242, Payload price. $354.00
Magnolia Microsystems 128K RAM Board. Add to your 48K H/Z-89 CPU memory for a total of 176K. For a limited time receive CP/M Plus (CP/M3), a $350.00 value, at no additional charge. Payload price $446.00
Magnolia Microsystems Memory Upgrade Card. Installs in your H-89 to increase memory from 48K to 64K. Payload price $68.50
Magnolia Microsystems SASI Bus I/O. Use to connect Winchester disk drives to your H-89. Also has three Serial Ports. Payload $225.00

UPGRADE ACCESSORIES FOR Z-100 and Z-150 SERIES COMPUTERS

Color Graphics Upgrade. All memory chips (16) required to update Z-100 Series computer for color. Installation instructions included ..... $94.50 Complete.
Memory Upgrade. All memory chips (9) required to upgrade Z-100 computers from 128 to 192 KB RAM. Upgrades Z-150 computers 64K additional for each kit ordered up to 320K maximum. Installation instructions included $53.25
UCI Memory Upgrade Card. 256K dynamic RAM installed. Empty sockets for an additional 256K dynamic RAM. We recommend this one highly. Retail $795.00, Payload $620.00
C.D.R. Systems Z-100 Speed Module. Run your Computer at 7.5 MHz instead of 5.0 MHz. Installs easily with no soldering. Externally switchable between Speed mode and Normal mode. Payload Priced $48.00
Second Internal Disk Drive. We can supply a matching drive to update your single drive Zenith computer to a two drive unit. Please let us know what you have and we will quote the Payload price.

EIGHT INCH DISK DRIVE SYSTEM

Two Tandon 8" TM848-2 Thinline, Double Density, Double Sided Drives in Deluxe Cabinet with Heavy Duty Power Supply and Cooling Fan, 2.4 MEG Storage Total. Complete with cable. Ready for H/Z-89 with Magnolia DD Controller or any H/Z-100 computer. Payload $1027.00

INTERNAL DUAL DRIVE SETUPS FOR THE H/Z-89 COMPUTER

Includes two half height double sided disk drives and all hardware and connectors required to mount inside your H-89. Also included is a steel mounting shield/case.
National JA551-2 Setup, 48 TPI, 6 MS seek, 640K total . . . $392.00
Mitsubishi 4851 Setup, 48 TPI, 6 MS seek, 640K total . . . $418.00
Mitsubishi 4853 Setup, 96 TPI, 3 MS seek, 1280K total . . . $479.00

WINCHESTER HARD DISK SYSTEMS

We can now supply complete hard disk systems for Zenith computers or separate components. Please write or call for the very best systems at low Payload prices.

HALF HEIGHT DISK DRIVES

Notice to clubs and quantity users. We can supply the following in case lots of 10 or 12 at reduced prices.

Tandon TM 848-2	8"	DS, DD, 48 TPI	$396.00
National JA551-2	5.25"	DS, DD, 48 TPI	$174.00
Mitsubishi M4851	5.25"	DS, DD, 48 TPI	$192.00
Mitsubishi M4853	5.25"	DS, DD, 96 TPI	$218.00

POWER SUPPLIES AND CASES FOR DISK DRIVES

For standard and half height disk drives. Rugged steel construction with heavy duty power supplies. Purchase with drives and we will configure and install drives in case for you.

Single	5.25" unit for standard drive	$ 68.00
Dual	5.25" unit, specify Standard or Half Height. . .	$ 92.00
Dual	8" unit with fan, for thinline drives	$225.00
Face plate for single drive in dual case		$ 8.00
For Winchester Drive, 5.25"		$225.00

Your satisfaction is guaranteed. All hardware carries a 90 day warranty. Prices listed have been discounted for payment with order. VISA and MASTERCARD orders please add 3%. Add $5.00 to all orders for handling and shipping, we pay the balance. Call us, we will be happy to answer your questions. THANK YOU.

VISA

PAYLOAD COMPUTER SERVICES
CHARLES C. PORTER, Proprietor
15006 SUN HARBOR, HOUSTON, TEXAS 77062
PHONE (713) 486–0687

MasterCard

Circle #147 on Reader Service Card

Hugh Kenner's Book for the Z100 Beginner

Heath/Zenith Z-100 User's Guide *will get you up and running, and even help you choose software.*

Mark Goodkin

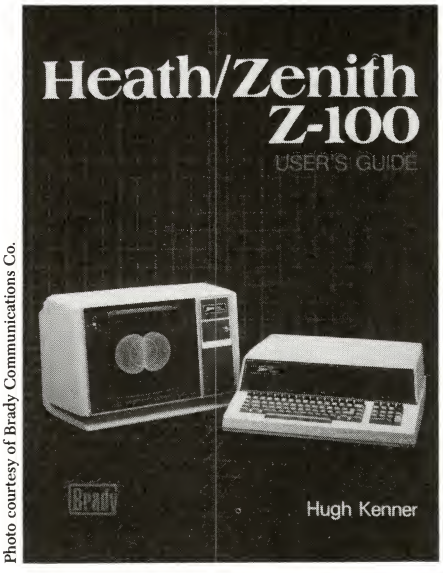


Photo courtesy of Brady Communications Co.

Heath/Zenith Z100 User's Guide
by Hugh Kenner
1984, Brady Communications Co., Inc.,
Bowie, MD
256 pages, paperback, $15.95

When I bought my first motorcycle several years ago, all I got with it was an instruction manual on how to repair it. I wish there had been one giving me tips on how to *ride* it—as well as something to tell me what kinds of accessories were available for it.

Computer users have tended to be in the same boat. But there is good news for computer novices who either own or contemplate owning the low-profile H/Z110 or the all-in-one H/Z120. Hugh Kenner has recently come out with his *Heath/Zenith Z-100 User's Guide.*

According to Kenner, his book is not intended to replace the '100's instruction manual, but to supplement it. The book basically does two things. First, it gets the user punching the computer keys sooner than the instruction manual would. Second, it goes beyond the manual by introducing the novice to different kinds of software for the '100.

Kenner's book also goes beyond the manual in at least two more ways: by helping the user make decisions about hardware and software; and by offering dos and don'ts on different aspects of the '100 and the software.

As a newcomer, myself, to the '100—and, for that matter, to the personal computer—I'm evaluating this book from the standpoint of the beginner towards whom Kenner gears his book. The book gave me a clear understanding of the '100 and its software.

Operating systems and hardware

Heath/Zenith Z-100 User's Guide opens up with a brief and yet solid introduction defining computer hardware and software. It's good in its down-to-earth definitions of these terms: hardware is "the heavy stuff you bring home in cartons," while software is "packaged thinking." The book's main body reflects this division into hardware and software.

Chapter 1, titled "Fundamentals," gives an overview of the important aspects of '100 hardware. It defines disks and disk drives, and describes the kinds of each used on the '100. Kenner then introduces the idea of color and color monitors. I didn't see, however, anything in this book explaining the uses of color or how to decide whether you really need a color monitor. Next, Kenner defines the computer terms bit, byte, and memory, relating each term to the other two.

Then he tells about operating systems, listing three different ones which can operate on the '100. They are the Zenith Disk Operating System (Z-DOS), CP/M-85 (8 bits), and CP/M-86 (16 bits). And he also discusses one which will operate on the '100 soon, the Heath Disk Operating System (HDOS). The book, however, concentrates only on two of these systems, Z-DOS and CP/M-85, in order to avoid repetition in explaining the similar features of each system.

After his description of the operating systems, Kenner shows how to begin operating the '100. He starts out with the more basic points such as booting. Moving to more complex ideas such as defaults, he shows how drive "A" remains the active drive until you specify otherwise. And he explains how to look up the directory of files on the disk. (It would have been helpful, however, if he had shown how to stop the list before it scrolled off the screen, although he does show you later on.) The first chapter ends by looking at disks and the '100 keyboard.

The next two chapters introduce the reader to Z-DOS and CP/M-85. Kenner describes the different features of each system, while giving you hands-on experience by running you through a series of steps. They include booting, formatting and copying disks, looking up files, and working with batch commands. Each step is stated simply and explained well.

But I did notice a problem with these two chapters. Every heading and subheading had a similar typeface. So I found it difficult sometimes to tell the difference. In fact, the entire book seems to suffer from this problem.

Chapter 4, only a few pages long, goes into a third operating system—HDOS—and yet just briefly because, as he explains, "certain things are common to all systems." In other words, he avoids repeating the same things about each operating system.

Programming and software

Part II, "Programming and Software," begins with Chapter 5, titled "1300 Heretical Words About Software." This chapter discusses the creation and refinement of software programs, making three points. First, don't expect the software program you buy from a company to be completely bug-free. A developer of software simply cannot predict *everything* that a user will do with a program.

And this fact of hidden bugs in the program leads to his second point: You, the user, can take part in debugging a program. Should you come across a bug—and he says that "odds are hugely against it"—you can call the vendor to fix it.

His final point in Chapter 5 is the "heretical" one (although probably not so for readers of *Sextant*). He advises that, when shopping around for software, you ought to first check out the "cottage-industry" software. It's cheaper, and you'll usually get better service. But you may have to go for the big commercial package, he writes, if that's the only software you can find to do a particular job you want.

The next two chapters introduce the reader to several computer languages

which can be used on the '100. Chapter 6, "The Gist of BASIC," presents four short lessons on BASIC, each lesson harder than the one before it. Kenner stresses that this chapter doesn't replace an instruction manual in BASIC. It merely gets the beginner off to a quicker start.

The thing that struck me the most in Kenner's instruction on BASIC is his advice on putting together a BASIC program. He says *not* to start figuring out the program from its beginning to ending. Instead, he says to "begin with the gist of the problem," and develop the program from that point.

For example, the problem in lesson two is to find the probability that at least two people among a specified number have the same birthday. The formula of the problem is a calculation involving the multiplication of a long series of fractions: (365/365)*(364/365)* ... *((366–P)/365); where "P" is the number of people specified. The denominators of the fractions stay the same, but the numerators drop by increments of one.

He shows that the computer can calculate this equation in BASIC through a FOR ... NEXT ... loop, counting backwards from 365. The first time the computer passes through the loop, it will come up with the first fraction in the equation. A second pass through the loop has the computer derive the second fraction (the numerator has dropped by one) and multiply it by the first fraction.

A third pass derives the next fraction and multiplies it by the product of the first two fractions. The computer continues to pass through the loops until it has multiplied the last fraction in the equation, (366—P)/365, by the product of the fractions preceding it. Kenner says that once you have come up with the gist of this program, the FOR ... NEXT ... loop, you can develop the rest of the program.

I do, however, have a couple of quibbles about Chapter 6. First, it's hard to distinguish the commas from the periods in the programs. Second, Kenner, waits until the last lesson to explain that BASIC doesn't care about the spacing surrounding signs such as "+" or "=". In the meantime, I was tempted to use a magnifying glass to make out the spacing.

The rest of Part II tells you what kinds of software are available for your '100. It also helps you to decide which ones are best for your needs by describing each of their virtues and limitations.

Chapter 7 explores the uses, strengths, and weaknesses of software languages such as BASIC, Pascal, COBOL, and Forth. Kenner spends most of his time here contrasting the formlessness of BASIC (with all of its GOTOs) to the strict order of Pascal. He thinks highly of Pascal for its structure but says that Structured BASIC is a good stepping stone from BASIC to Pascal. The rest of the chapter skims over the other languages, telling you, among other things, where you can buy them.

Chapter 8, "Editors and Word-Processors," covers, as the title suggests, different word processor packages such as PIE and TEXT (from The Software Toolworks), WatchWord (from S & K Technology, Inc.), and Perfect Writer (from Perfect Software, Inc.). Kenner doesn't give step-by-step instruction on how to use these programs. Instead, he gives several characteristics of each—pointing out PIE/TEXT's simplicity, WatchWord's versatility, and Perfect Writer's capability to predefine print formats. (The other editors that Kenner discusses are EDT, WordStar, PeachText, and Spellbinder.) Here, as elsewhere, his comments do not substitute for a manual—but they serve as a good introduction to the product.

Chapter 9 is titled "Dealing with Data Bases." It defines a data base, shows its practical value, and compares two of those used most commonly: Query![2] (from Hoyle & Hoyle Software) and dBASE II (from Ashton-Tate). Comparisons are made on the following criteria: cost, reliability, limitations, applications, and ease of use. I like Kenner's comparison of a library's card catalogue to a data base. With a card catalogue, "each new searching field [author, subject, and title] needs a new set of cards," while "the data base program can use any field as the sorting or searching field." This chapter also gives you a good idea of which data base program is best for your needs.

Chapter 10 is about spreadsheets. It defines them and points out their uses. This chapter is quite brief and gives only a general explanation of Multiplan.

Chapter 11 explains how you and your '100 can exchange information with other people and their computers. It shows how the telephone ties you to bulletin boards, individual computers, and data bases.

The last chapter looks at graphics. It thoroughly explains how a particular configuration of dots on the '100's screen lights up to create a certain graphic design. It describes some of the different kinds of graphs which you can create—from simple geometric figures to designs of complex objects such as the Space Shuttle—and tells how to create them.

Introduced in this chapter as well is the use of color in graphics. Here, Kenner lists the colors available for the '100 and offers some simple instruction on displaying one color or more—such as a white design on a blue screen.

Incidentals

The book's glossary is clearly written, using boldface type to distinguish the word to be defined from its definition.

Professional Graphics at Practical Prices

Add our Imaginator™ graphics upgrade card to your H/Z-19 terminal or H/Z-89 computer. It's quick and easy. You gain intelligent, highly efficient graphic display capabilities proven in countless Heath/Zenith terminal and computer updates.

Check Imaginator's special features:

- ☐ High resolution 504 x 247
- ☐ Accessible through any high level language FORTRAN, Pascal, BASIC, etc.
- ☐ Onboard microcomputer eliminates processing load on host
- ☐ Source code available
- ☐ Rich graphics instruction set
- ☐ Mix text and graphics
- ☐ Tektronix® 4010-4014 compatibility option with GIN mode
- ☐ Comprehensive documentation includes numerous examples
- ☐ Fully buffered for asynchronous operation
- ☐ All original H/Z features remain intact

Now check Imaginator's low cost:

- ☐ Assembled complete $445.
- ☐ Kits from $215.

Also ask us about our Imaginator 2 upgrade for H/Z-29 terminals.

Call or write us today for additional information.

CLEVELAND CODONICS, INC.
18001 Englewood • Cleveland, OH 44130 (216) 243-1198

Your H-19 Can Do This...

IMAGINATOR

Circle #112 on Reader Service Card

Tektronix® Registered trademark of Tektronix, Inc.
Codonics and Imaginator are trademarks of Cleveland Codonics, Inc.

This makes it easy to find a word. Also, the glossary's definitions are easy to understand.

There's a thorough index: 13 pages with two columns on each page. (I was somewhat confused by the list of symbols preceding the index. It doesn't have a heading to identify it, leaving me briefly staring at a column of words until something clicked inside. Such confusion could have been avoided with a heading such as "symbols".)

The graphics in the book are nicely done. Many of the pictures are well "worth a thousand words." The full-page diagram in Chapter 2 on formatting bootable and non-bootable disks is particularly helpful. It reduces a tedious description of a formatted disk to a diagram of the kinds of files on each disk.

I felt that the picture of the '100 keyboard in Chapter 1, though realistic, was too small. Perhaps something in the neighborhood of a half page would make more sense.

Steering the new user

Kenner takes care to offer useful tips to the new user. For instance, in Chapter 1, he gives advice on how to protect your disks when you're not using them. He provides a thorough list of don'ts—from bending them to putting them "anywhere near anything magnetic."

Kenner also helps the reader make many important decisions about computer hardware and software, by pointing out the strengths and weaknesses of various kinds of products. For example, he discusses which kind of printer you might buy—daisywheel or dot matrix. He explains the trade-offs in looking at a dot-matrix printer, which is fast but produces relatively poor print quality for ordinary text. A daisywheel, on the other hand, is slow but gives good manuscript-quality print. (Incidentally, though, he never really emphasizes the value of dot matrix in graphics.)

In another example, he helps the user match the right word-processing program to fit a particular need. This kind of tip no instruction manual will ever provide. To repeat, however, this book isn't intended to replace an instruction manual, but to augment it. Ideally, it should be read before the manual.

Kenner does an excellent job explaining the concepts of hardware and software. He loves to use analogies, which I find good because it keys me into something unfamiliar by first looking at something familiar which is similar to it. For instance, he writes that a central-processing-unit chip "looks like a stick of black chewing gum with centipede legs." How's that for visualization?

Or how about the operating system? He writes, "You might compare it to your own nervous system." He then thoroughly explains his analogy.

He also uses stories to explain ideas. The one that struck me the most is his explanation of hardware and software. Once upon a time, the old coffee machines had a lot of buttons—including CUP and LID. If you forgot to push CUP, you were out of luck. The later models gave you a cup and lid automatically. In other words, they were *programmed* to dispense a cup and lid. Like the coffee machine, your computer, or *hardware*, cannot think for itself. It needs a program, or *software,* to do something such as calculate a math equation.

In fact, I feel that his stories add human interest—which helped to hold my attention. Kenner uses these kinds of explanations throughout his book.

All in all, the book is well thought out, covering the important topics of the '100. In addition, it's indispensable for the novice, being the only one of its kind on the market for the '100. And it's readable. Not only was the book clearly written, but the computer terms were clearly defined.

We shouldn't be surprised, though, that Mr. Kenner writes well. The fact that he's an avid Z100 user is not his only qualification to write this book. He is a professor of English at Johns Hopkins University in Baltimore, Maryland, and has written several books in his field.

If the book has some shortcomings, they're dwarfed by its strengths. After you've invested a few grand in a '100, it makes sense to invest another $15.95 in a user's guide. It will help you get much more mileage out of your personal computer.

Get *more news* about your Heath or Zenith computer system *more often*—AND get a *FREE* directory of over 350 sources of support for your computer.


BUSS #79
The Independent Newsletter of Heath Co. Computers

Subscribe to **Buss: The Independent Newsletter of Heath Co. Computers**
and also receive the directory *at no additional cost*

You can't get more frequent news about your system than with a subscription to *Buss*. Twenty times a year, in each and every issue, you'll find out what other users and independent suppliers have to say about:

- ☐ H8
- ☐ H/Z89
- ☐ Z100
- ☐ compatible hardware
- ☐ printers
- ☐ terminals
- ☐ disks and drives
- ☐ compatible software
- ☐ business applications
- ☐ home/hobby applications
- ☐ games
- ☐ graphics
- ☐ utilities
- ☐ Z-DOS
- ☐ HDOS
- ☐ CP/M

Twenty times a year, you'll get a complete report on the latest news about your Heath/Zenith system—delivered directly to you by first class mail. *Buss* helps you:

- Learn what new products are being developed, and where you can get them.
- Benefit from the advice of users when you write in to *Buss*'s "Requests for Assistance" column.
- Save money when you add to your system! *Buss* offers a free "For Sale" listing for Heath/Zenith users where you can pick up a bargain.
- Avoid bugs in your system as you read how other users have solved similar problems with their computers.
- Discover news and events in the Heath/Zenith community that affect you.
- Meet other users in your area through the "Local Club Notes" section of *Buss*.
- Get unbiased coverage of your system from the oldest independent newsletter for Heath/Zenith users. *Buss* is committed to supplying more information exclusively for Heath/Zenith users, more often than any other publication!

And, as a free bonus with your *Buss* subscription, you'll receive your FREE copy of *The Buss Directory*—a reference tool which provides you with information on over 350 sources of independent support for your Heath/Zenith system.

The Buss Directory is a $12.50 value—and it's yours FREE with your subscription to *Buss!* We'll send it to you by first class mail as soon as we receive your subscription order.

Order your subscription today, and start getting MORE NEWS, MORE OFTEN about your Heath/Zenith system!

Use this coupon, and mail your order today to:
Buss, Dept. S14
716 E Street, S.E.
Washington, D.C. 20003
202/544-0900

Your satisfaction with *Buss* is completely guaranteed. If at any time you are not satisfied, your money will be refunded—in full.

Yes! Sign me up for *Buss*, and send me my free copy of *The Buss Directory*!

I want:
- ☐ 30 issues for $38 ($55 overseas)
- ☐ 20 issues for $28 ($40 overseas)
- ☐ 10 issues for $18 ($24 overseas)

- ☐ Payment enclosed (checks must be in U.S. dollars payable on a U.S. bank).
- ☐ Bill me.
- ☐ Charge my: ☐ Visa ☐ MasterCard

Name_____
Address _____
_____
_____

Card # _____
Expiration date_____
(MC code #_____)

Buss, Dept. S14, 716 E Street S.E., Washington DC 20003

Circle #107 on Reader Service Card

Circle #164 on Reader Service Card

Put HDOS Commands on Disk

This program will store commands and feed them into HDOS's type-ahead buffer.

Roy Coleman

In using a computer, we frequently find ourselves giving the computer basically the same series of commands today that we gave it yesterday, and that we will give it tomorrow. We call up an assembler, perhaps, assemble a file, and print the original .ASM file.

We can do this automatically under both the CP/M operating system and the Zenith Disk Operating System (Z-DOS). We simply set up a disk file containing the commands we would otherwise type into the computer. The operating system can then receive the commands from disk. Under CP/M, the file of commands is called a SUBMIT file. Under Z-DOS, this is called batch processing, and the disk file a batch file.

Under either CP/M or Z-DOS, the operating system performs whatever is called for in the SUBMIT file. An example might be running the assembler. All the while, the SUBMIT process keeps track of how far it has gone through the disk file. When control is returned to the operating system, the next line is executed.

But what about the Heath Disk Operating System (HDOS)? HDOS does not yet come with a SUBMIT capability.

The type-ahead buffer

One way to emulate some of the power of a SUBMIT file is possible, however, because of HDOS's type-ahead buffer. This buffer can hold about 100 characters if for any reason keyboard input cannot be acted upon. (You can ask for a CATalog while HDOS is busy mounting a disk, for instance.)

A program can be written, then, that will let HDOS read a SUBMIT file; the program will send all the commands to the type-ahead buffer. This will give you a limited SUBMIT capability. (The Heath Users' Group—HUG—offers a limited SUBMIT utility on its Disk VII Utilities Package.)

The main limitation of relying on the type-ahead buffer is that it has no way of knowing whether a program is in charge or whether you are at the HDOS level. True batch processing can turn over control to your program, then resume execution of the command file after the program returns control to the operating system. Using the type-ahead buffer, if the first line of your file called up your editor or word processor, for example, then the remaining lines would be input to the word processor.

That might be all right if you stayed in your editor's command mode. Then you could give those commands to the editor. But if you had to go into text-entry mode, then those lines would just be treated as ordinary text.

That is the first limitation to using the type-ahead buffer under HDOS to get a limited SUBMIT capability. This seems to be inherent in a simple use of the type-ahead buffer. But if a particular series of commands is to be executed several times, a SUBMIT-type program works quite well under HDOS.

The idea of using the type-ahead buffer has been around for some time. Several articles in HUG's *REMark* magazine describe the use of the type-ahead buffer. (See *REMark* #7, 1979, p. 23, "BASIC Ideas," by Allan H. Moser; #13, December 1980, "Editor's Note," pp. 26-27, and "Another Pointer to the Type-Ahead Buffer," by J. H. Gold, pp. 24-26; #17, May 1981, "Correction to Previous Note," p. 20; and #19, August 1981, "SUBMIT Update," p. 3.)

Unfortunately, such a SUBMIT-type program is useful only if the commands are always going to be exactly the same. You have to specify devices and file names in the command file itself.

To get around this limitation, I wrote Command Processor, the program shown in Listing 1. This program works in much the same way as most other HDOS SUBMIT programs in that it places a line or lines in the type-ahead buffer.

It also provides the user with a way of entering a file or device specification on the HDOS command line and having that file or device name included in the series of commands to be carried out.

I chose to write Command Processor somewhat differently from what you might expect for a SUBMIT capability. The commands are not stored in a separate disk file. Instead, they are incorporated into the Command Processor program itself. To change the commands, a line of the program is changed. This may seem to be somewhat awkward. But it produces shorter, less complex code by doing all the work in the program. Assembled, Command Processor takes up only two sectors on disk.

Some examples

A couple of examples should provide you with an idea of what can be accomplished with this program.

First, suppose that you want to load Benton Harbor BASIC and then load and run a particular program. I normally do this by entering BASIC, and, at the * prompt, entering CHAIN "PROGNAME". (I use CHAIN to combine both the OLD "PROGNAME" and RUN commands.)

By using Command Processor exactly as shown in Listing 1, naming the resulting .ABS file C.ABS, you could simply enter C PROGNAME and BASIC would automatically be loaded and chained to PROGNAME.

To change to a different series of commands, you only need to appropriately change the line marked COMAND. (See below, where I refer to

Listing 1. Command Processor. This program works by placing commands in the type-ahead buffer. The commands are incorporated in the program itself, and are changed by changing the COMAND line in section 13.

```
          TITLE   'Command Processor'                              1
          STL     'Externals, Definitions and ID'
*         AUTHOR  -  Roy Coleman
*         DATE    -  October 27, 1983
FALSE     EQU     0               LOGICAL FALSE
.EXIT     EQU     0               HDOS EXIT ROUTINE
.SCIN     EQU     1               HDOS KEYBOARD INPUT ROUTINE
TRUE      EQU     1               LOGICAL TRUE
O.LC      EQU     2               LINE COUNTER
O.QTPT    EQU     6               QUEUE TAIL POINTER
O.QHPT    EQU     8               QUEUE HEAD POINTER
O.BSTPT   EQU     10              BUFFER START POINTER
O.BENPT   EQU     12              BUFFER END POINTER
USERFWA   EQU     042200A         USER WORKING ADDRESS
S.DLINK   EQU     040346A         BUFFER LINK
$TYPTX    EQU     031136A         TYPE TEXT ROM ADDRESS
$CDEHL    EQU     030216A         COMPARE DE AND HL ROM ADDRESS
$HLIHL    EQU     030211A         LOAD HL THROUGH HL ROM ADDRESS
          STL     'Get Name of File to Process'
          EJECT
          ORG     USERFWA                                          →
```

that line simply as the COMAND line.) Then this program could be used to load and run the assembler, ASM, from a disk other than SY0: and to automatically enter the ASM command line. By assembling this program and naming it A.ABS on SY0:, the user could just enter A SY1:F1NAME and A.ABS would load the assembler from SY1: and assemble SY1:F1NAME.ASM into SY1:F1NAME.ABS. It would optionally produce a listing file on SY1: or some other device such as a printer. Entering A SY2:F2NAME would do the same from SY2: using SY2:F2NAME.

My third example involves the spooler program which was published in *Sextant* #6, Summer 1983 ("Inside a Spooler," by A. E. Dessler). In order to use the spooler as published, you need to do several things; they include setting SP:=TT:, entering the file name, and ending with a control-D. By properly setting up Command Processor, all that you need to do is enter SPOOL FILENAME and the rest takes place automatically.

Command Processor obtains a file name from either the HDOS command line or by requesting it from the keyboard. Whenever it encounters a 377Q in the COMAND line, it places the file name, including device, into the type-ahead buffer. Additionally, if this program encounters a 376Q in the COMAND line, it will insert just the device specification (if any) from the file name entered at the beginning of execution.

In the first example given above, the COMAND line is:

```
COMAND DB 'BASIC',12Q,'CHAIN "',377Q,
'"',12Q,0
```

The elements of the line are delimited by commas. The 12Qs go where you would normally hit <RETURN>. At execution, the file name would replace the 377Q. The 0 at the end is a terminating byte and *must* be the last byte of the COMAND line since Command Processor keeps placing things into the type-ahead buffer until it finds a 0. (No 0 and it will just move to the next line looking for more.)

For the second example, COMAND should be as follows:

```
COMAND DB 376Q,'ASM',12Q,377Q,',LP:=',
377Q,12Q,0
```

Again the 12Qs are <RETURN>s and the line is terminated by a 0. The 376Q indicates to Command Processor that it should get a device specification from the command line and place it at this point. If no device was initially specified, SY0: is assumed. The 377Qs are where the file name is placed. The LP: tells the assembler to send the listing file to the LP: device.

Assuming the spooler is loaded, the COMAND line for the third example should be:

```
COMAND DB'COPY SP:=TT:',12Q,377Q,04Q,0
```

Again the 12Qs and 0 at the end are as in the previous examples. The file name entered on the HDOS command line would be placed where the 377Q is

```
*   CHECK COMMAND LINE FOR NAME                                    2
START   LXI     H,0             SET UP HL AS BASE REGISTER
        DAD     SP              ADD STACK POINTER STARTING ADDRESS
        MOV     A,L             CHECK STACK POINTER
        CPI     200Q            HAS IT MOVED FROM BASE?
        JZ      GETNAM          NO, NO NAME ON COMMAND LINE, GET ONE
        LXI     D,BUFFER        GET ADDRESS OF BUFFER FOR NAME
MOVIT   MOV     A,M             MOVE CHAR FROM STACK TO REG A
        INX     H               POINT TO NEXT CHAR ON COMMAND LINE
        CPI     ' '             WAS IT A SPACE?
        JE      MOVIT           YES, GO BACK FOR MORE
        CPI     ':'             WAS IT A : INDICATING DEVICE OTHER THAN SY0:
        JNE     MOVIT1          NO, JUMP AROUND FLAG SET
        CALL    GETDEV          REFILL DEVICE BUFFER WITH CORRECT DEVICE
MOVIT1  CPI     0               WAS IT A NULL (END OF LINE)
        JE      MOVITND         YES, GO TO END OF LINE ROUTINE
        STAX    D               SAVE CHAR IN FILE NAME BUFFER
        INX     D               WHERE TO PUT NEXT CHAR
        JMP     MOVIT           GO BACK FOR MORE
MOVITND MVI     A,0             FILL LAST SPACE WITH NULL
        STAX    D               SAVE IT
        JMP     MOVCMD          GO TO FILLING TYPE-AHEAD BUFFER
*   IF NO NAME ON COMMAND LINE, ASK FOR FILE NAME
GETNAM  LXI     H,BUFFER        GET STARTING ADDRESS OF FILE NAME BUFFER
        CALL    $TYPTX
        DB      'Enter the name of the file you want processed - ',200Q
RDLOOP  SCALL   .SCIN           GET A CHARACTER FROM COMMAND LINE
        JC      RDLOOP          NO CHARACTER, GO BACK
        CPI     ' '             WAS IT A SPACE?
        JE      RDLOOP          IGNORE IT AND GO BACK FOR MORE
        CPI     12Q             WAS IT A CARRIAGE RETURN (NEW LINE)?
        JE      NDLOOP          END OF FILE NAME ENCOUNTERED
        MOV     M,A             SAVE CHARACTER IN BUFFER
        INX     H               GO TO NEXT POSITION IN BUFFER
        CPI     ':'             HAS A DEVICE BEEN ENTERED
        JNE     RDLOOP          NO, GO BACK FOR MORE
        CALL    GETDEV          REFILL DEVICE BUFFER
        JMP     RDLOOP          GO BACK FOR MORE
NDLOOP  MVI     M,0             TERMINATE WITH A NULL (FALL THROUGH TO MOVCMD)
        STL     'Put Command Lines Into the Type-Ahead Buffer'
        EJECT

MOVCMD  CALL    SETUP           SET POINTER ADDRESSES                  3
        LXI     H,COMAND        LOAD STARTING ADDRESS OF COMMAND LINES
MOVMEM  MOV     A,M             GET BYTE FROM COMAND
        INX     H               POINT TO NEXT BYTE
        ORA     A               WAS IT LAST BYTE? (A NULL)
        JZ      XIT             IF YES, EXIT
        CPI     377Q            PUT FILE NAME HERE?
        JE      MOVNAME
        CPI     376Q            PUT DEVICE HERE?
        JE      MOVDEV
        STA     SAVCHR          IF NO, SAVE THE CHARACTER
        CALL    PUTIN           PUT THE CHARACTER IN THE TYPE-AHEAD BUFFER
        JMP     MOVMEM          GO BACK FOR MORE

*    EXIT ROUTINES                                                  4
XIT     LDA     SAVCHR          NORMAL EXIT, WAS LAST CHR A LF?
        CPI     12Q               IF YES, SKIP, IF NO, INCREMENT
        CNE     INCLP             LINE COUNTER IN TYPE AHEAD BUFFER
XIT1    XRA     A               NORMAL AND ABORT EXIT
        SCALL   .EXIT

*    MOVE THE FILE NAME INTO THE COMMAND LINE                       5
MOVNAME PUSH    H               SAVE COMMAND LINE ADDRESS
        LXI     H,BUFFER        GET FILE NAME ADDRESS
MOVNAM1 MOV     A,M             GET CHAR FROM FILE NAME
        INX     H               POINT TO NEXT CHAR
        ORA     A               LAST CHAR?  (A NULL)
        JE      MOVNAM2         YES, CLEAN UP STACK AND GO BACK
        STA     SAVCHR          SAVE THE CHAR TO SHOVE IN TYPE AHEAD BUFFER
        CALL    PUTIN           PUT IT IN BUFFER
        JMP     MOVNAM1         GO BACK FOR MORE
MOVNAM2 POP     H               CLEAN UP STACK
        JMP     MOVMEM          GO BACK TO COMMAND LINE PROCESSOR

*    MOVE THE DEVICE NAME INTO PLACE BUT ****ONLY**** IF JUST DEVICE NEEDED
MOVDEV  PUSH    H               SAVE COMMAND LINE ADDRESS              6
        LXI     H,FDEV          GET ADDRESS OF DEVICE BUFFER
MOVDEV1 MOV     A,M             GET CHAR FROM FILE NAME BUFFER
        STA     SAVCHR          SAVE FIRST THREE CHARACTERS AS DEVICE
        CALL    PUTIN           SAVE IN TYPE-AHEAD BUFFER
        INX     H               GET ADDRESS OF NEXT CHAR
        LDA     SAVCHR          GET LAST CHAR BACK
        CPI     ':'             WAS IT DEVICE DELIMITER?
        JNE     MOVDEV1          NO, GO BACK AGAIN FOR MORE
        POP     H               RESTORE STACK
        JMP     MOVMEM          GO BACK FOR MORE

*    IF TYPE-AHEAD BUFFER IS FULL, PRINT MESSAGE AND EXIT (ENTERED FROM PUTIN)
BUFFUL  POP     H               CLEAN UP STACK                         7
        CALL    $TYPTX
        DB      07,'Buffer full, job abandoned!',212Q                  →
```

```
        JMP     XIT1            ABORT EXIT
        STL     'Subroutines'
        EJECT

*    SET ALL POINTERS AND ADDRESSES                                    8
SETUP   LHLD    S.DLINK         GET ADDRESS OF HIGHDAT
        XCHG                    PUT ADDRESS OF HIGHDAT IN DE
        LXI     H,0.LC
        DAD     D
        SHLD    LC              SAVE LINE COUNTER
        LXI     H,0.QTPT
        DAD     D
        SHLD    QTPT            SAVE TAIL POINTERADDRESS
        LXI     H,0.QHPT
        DAD     D
        SHLD    QHPT            SAVE HEAD POINTER ADDRESS
        LXI     H,0.BSTPT
        DAD     D
        SHLD    BSTPT           SAVE TYPE-AHEAD STARTING POINTER ADDRESS
        LXI     H,0.BENPT
        DAD     D
        SHLD    BENPT           SAVE TYPE-AHEAD ENDING POINTER ADDRESS
        RET

*    INSERT A CHARACTER INTO THE TYPE-AHEAD BUFFER                     9
PUTIN   PUSH    H               SAVE TABLE POINTER
        LHLD    QTPT            GET ADDRESS OF TAIL POINTER
        CALL    $HLIHL          GET TAIL BYTE
        LDA     SAVCHR          GET CHARACTER TO PUT AT TAIL BYTE
        MOV     M,A             PUT IT THERE
        CPI     12Q             WAS IT A CARRIAGE RETURN?
        CZ      INCLP           YES, INCREMENT LINE COUNTER
        INX     H               INCREMENT TO NEXT POSITION IN QUEUE
        XCHG                    PUT NEXT POSTION ADDRESS IN DE
        LHLD    BENPT           GET ADDRESS OF END POINTER
        CALL    $HLIHL          GET ITS CONTENTS
        CALL    $CDEHL          COMPARE IT TO THE END POINTER
        CZ      SETHEAD         IF AT END, RESET POINTER TO START
        LHLD    QTPT            UPDATE TAIL POINTER
        MOV     M,E             GET ADDRESS AND CONTENTS OF TAIL POINTER
        INX     H                 AND SAVE IT
        MOV     M,D
        LHLD    QHPT            GET HEAD POINTER ADDRESS
        CALL    $HLIHL          GET CONTENTS OF HEAD POINTER ADDRESS
        CALL    $CDEHL          IS BUFFER FULL?
        JZ      BUFFUL          YES, PRINT ERROR MESSAGE AND ABORT
        POP     H               RESTORE POINTERS
        RET

*    GET THE ADDRESS OF THE START OF THE TYPE-BUFFER IN DE            10
SETHEAD LHLD    BSTPT
        CALL    $HLIHL          PUT ADDRESS OF TYPE-AHEAD BUFFER IN HL
        XCHG
        RET

*    INCREMENT THE LINE COUNTER                                       11
INCLP   PUSH    H
        LHLD    LC              POINT TO LINE COUNTER
        INR     M               INCREMENT IT
        POP     H
        RET

*    FILL DEVICE NAME BUFFER                                          12
GETDEV  PUSH    PSW
        DCX     H               BACK UP TO START OF DEVICE NAME (3 CHAR)
        DCX     H
        DCX     H
        DCX     H
        MOV     A,M             PUT IT INTO FDEV BUFFER, ONE CHAR AT A TIME
        STA     FDEV
        INX     H
        MOV     A,M
        STA     FDEV+1
        INX     H
        MOV     A,M
        STA     FDEV+2
        INX     H
        INX     H
        POP     PSW
        RET
        STL     'Storage Area'
        EJECT

SAVCHR  DB      0                                                    13
FDEV    DB      'SY0:'
LC      DW      0
QTPT    DW      0
QHPT    DW      0
BSTPT   DW      0
BENPT   DW      0
COMAND  DB      'BASIC',12Q,'CHAIN "',377Q,'"',12Q,0
BUFFER  EQU     *
        END     START
```

located; the 04Q is the spooler's exit code.

The listing

The parts of the program in Listing 1 are numbered 1 through 13.

Lines in section "1" provide the necessary equates for the assembler.

Lines in "2" get the name of the file to be processed. The first part is similar to the process described by Pat Swayne in *REMark* #45 (October 1983, "Getting Started With Assembly Language"). It checks for a ":" indicating that some type of a device has been specified.

The first two lines in "3" set up the pointers and load the address of the beginning of the COMAND line. The remainder moves one byte at a time to the type-ahead buffer, checking for the end (i.e., the 0 byte) or for name (377Q) or device (376Q) insertion.

"4" provides for an orderly exit, incrementing the line counter if the last line did not end in a <RETURN>. (See the third example.)

"5" moves the name of the file specified (including the device) into the type-ahead buffer; while "6" moves only the device specification.

The lines in "7" abort the program if the type-ahead buffer fills. "8" initializes all of the pointers. "9" is the actual routine which puts one character into the type-ahead buffer and sets the pointers for the new length.

"10" gets the address of the start of the type-ahead buffer. "11" increments the line counter which tells HDOS how many lines are waiting to be processed.

Section "12" sets up SY0: as the default drive, and changes it *only* if a drive specification was entered on the command line. It detects the drive specification by the ":" and then places the three previous bytes in FDEV as the drive specification for this execution. (If no drive specification was entered, and a 376Q is encountered, FDEV will be entered as the default.)

FDEV can be set to anything. So if you always execute programs from DK1:, you can change FDEV to DK1:, eliminating the need to continually type DK1: on the command line.

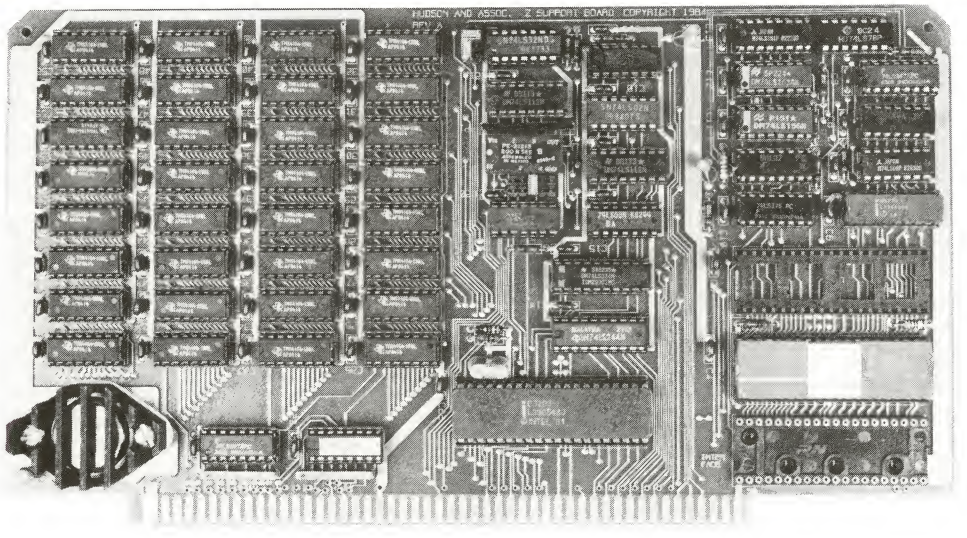
Finally, "13" contains the storage allocations. COMAND is the line containing the commands to be processed.

As noted above, the COMAND line must always be terminated with a zero. You may wish to avoid typing this in. You can have the program automatically end the COMAND with a 0 by adding the following line immediately after the COMAND line:

```
CMDEND  DB  0
```

As noted, this program assembles to an .ABS file only two sectors long. So it should be possible to assemble several versions of it. You will not have to use an excessive amount of disk space—and will ultimately save a considerable amount of command typing.

8087 SYSTEM SUPPORT



Z SUPPORT BOARD

The **Z SUPPORT BOARD** allows the user to add Intel's high performance 8087 high speed Numerical Data Processor and 256K bytes of memory to the Zenith Z100 computer system. This combination creates an extremely powerful CPU unit that is capable of the high speed number-crunching required in today's scientific and business applications.

- Easy installation of the Z Support board. Complete Technical Manual containing installation procedures, circuit description, theory of operation, and schematics.
- Operates with no operating system modification. Allows user to run both 8085 and 8086 code.
- Allows 8087 hardware interrupt for exception handling.
- OPTION 1 — 8087 circuitry fully assembled and tested — socketed for 256K upgrade. (note, memory circuity not tested)
- OPTION 2 — 256K of memory fully assembled and tested — socketed for 8087 upgrade. (note, 8087 circuitry not tested)
- OPTION 3 — 8087 circuitry and 256K of memory fully assembled and tested. (note, upgrade kits available)

OPTION 1	$595.00
OPTION 2	$595.00
OPTION 3	$995.00
TURBO PASCAL	$ 89.00

shipping and handling $10.00
Prepay or C.O.D. orders accepted (note, prices subject to change without notice)

HUDSON
a•n•d associates

P.O. Box 2957
Santa Clara, CA 95055-2957
Phone (408) 554-1316

Circle #138 on Reader Service Card

Heathkit Hackers at Home: A Survey of Hobbyists

Whether the computer helps with their hobby or is the hobby, these users are combining enjoyment and education.

David R. Felstul

Okay, so you finally bought a computer. And you've already used it to balance last month's check statement for the tenth time today. You also bought that printer that can handle all those different type fonts—but you don't feel like writing a letter to Aunt Gertrude in italics right now. So now what do you do with your computer? Well, how about using it as a *hobby*?

To give you just a few ideas of how your fellow *Sextant* readers use their Heath/Zenith computers as a hobby, we took a survey. We wrote to those people who indicated on their reader service cards (*Sextant* #9) that they used their Heath/Zenith computer primarily as a hobby. And what did we find out?

First of all, the definition of a computer hobby varied from respondent to respondent. Some respondents, of course, use their computers with a traditional hobby such as amateur radio. But for many, just using the computer is their hobby. They modify their computers, write software, or simply spend their time tinkering with them.

Regardless of how they use their computers for a hobby, almost all of the respondents feel that they have learned a lot from using them. This educational benefit is one of the two main reasons respondents gave for having a computer hobby. The other reason is enjoyment.

Many computer hobbyists enjoyed building their computer just as much as they enjoy tinkering with the assembled product. And although they didn't originally plan on it, most respondents soon found that the computer that they had built just for fun quickly became indispensable for all those little, everyday tasks that everyone faces, such as letter writing or record keeping.

What kind of person is a computer hobbyist?

Computer hobbyists often have a high degree of computer sophistication even before buying a Heath/Zenith microcomputer. Frank Davis, for example, used to work for the Boeing division which provides support to the National Aeronautics and Space Administration. At that time, he helped build a computer for NASA to use for ground support. Recently he decided that he wanted to get back into computer technology. He picked a Heath computer because it allowed him to build a "first class computer and get directly involved with the software."

Several other hobbyists also have had careers in the computing field. James Knerr is a computer consultant. Jack Fromkin has used computers in biomedical research. And another hobbyist, Jim Eilers, has used mainframe computers for more than 30 years to design petroleum plants. These people just can't seem to get enough of computers at work, but use them at home as well. And with a Heath/Zenith computer they can really get inside the computer and experiment with the hardware as well as the software.

These people just can't seem to get enough of computers at work, but use them at home as well.

This computer expertise shows in several areas. For instance, although we didn't specifically ask the question, many respondents told us which programming languages they used. On the average, each respondent knew three languages—with BASIC, assembler, and FORTRAN being the ones most frequently mentioned.

Nearly three-quarters of the hobbyists belong to either a local Heath Users' Group or the national one. Quite a few have joined both. Overall, they find the HUGs to be helpful—especially when it comes to buying software at a discount or getting technical advice, such as how to configure printers.

Knowing how valuable a HUG can be, one can only sympathize with Chuck Bialkowski, who lives in Saudi Arabia. He wrote that there was no possibility of his belonging to a local HUG since Zenith is blacklisted in Saudi Arabia. He does maintain ties with both the national and St. Paul-Minneapolis HUGs in the States, though.

The majority of the hobbyists, about three-fifths, own an H89. But the number who own an H/Z100, almost half of the survey respondents, is close and growing rapidly. Not as many hobbyists own an H8, fewer than one-tenth—although many previously did and then upgraded to an '89 or '100. (You may note that this adds up to more than 100%. That's because one-seventh or so of the hobbyists own more than one Heath/Zenith computer.)

How they got started

Many of the respondents reported that they have been building Heathkits for years. In fact, there seemed to be some kind of contest going for who had been building Heathkits the longest. For those of you who proudly announced 10 or 15 years, I'm sorry, but the winner seems to be Jack Fromkin of Los Angeles. He writes that he has been saving his pennies to buy electronic voltmeters, audio generators, and the like in Heathkit form since about 1948, only a year or so after the Heath Company started selling electronics kits. S. Henry Frankel, hailing from Wantagh, New York, also started building Heathkits around that time.

Thus, a lot of hobbyists had previously built stereo components and televisions. They now wanted to build the ultimate kit—a computer. Those who weren't such Heathkit "pros" often bought a Heathkit computer because of a National Technical Schools program. In NTS's Master Course in Microcomputers, the final project is to build a computer—the H89.

Both the experienced Heathkit builders and the NTS graduates said they liked building the computer from a kit because it gave them a better understanding of how the computer worked—instead of just blindly pushing buttons on the keyboard. Even people who had been using computers for years seemed to feel this need to know exactly how

their microcomputer worked.

There is also the enjoyment of building a complicated kit and having it come to life under your hands. In the words of Neil J. Adler from Suisun, California: "Let me tell you; it was quite exciting when I first plugged the H8 in, punched in the machine code on the front panel and saw the message 'Your H8 is up and running' on the LED display." No matter how exciting those first few moments are, however, turning the power on is just the beginning of the enjoyment of a computer hobby.

What does a computer hobbyist do?

As Tom Lufkin of Charleston, South Carolina, points out, "*Random House* defines a hobby as an activity or interest pursued for pleasure or relaxation and not as a main occupation." According to this definition, he feels that he qualifies as a computer hobbyist.

P. G. Manney from Medina, Ohio, would also easily qualify. Mr. Manney writes programs for relaxation and adds "Yes, my wife thinks I am weird too." (Tell your wife not to worry, Mr. Manney—there are a lot of hobbyists/programmers out there relaxing in front of a terminal.)

Why do so many hobbyists regard computer programming as relaxing? Jim Eilers seems to answer that quite well when he writes: "Most hobbies by definition should have an essential quality of creativeness; this is what I believe a computer is good for. I have several friends who program on their microcomputers just for the fun of creating a program. Somehow to them this is a creative act, when you see your construction take wing after you first hit the RUN button. One is never completely satisfied with the effort of creation. This one friend says that to create a program is fun, but to create an 'elegant' program is real class and worth striving for."

Jack Fromkin expresses much the same feeling when he states that "programming is addictive." And Douglas R. Roberts of New Gloucester, Maine, also uses his computer for "just tinkerin' around."

Despite their protestations that they really are only doing it for fun, almost every hobbyist we heard from seemed to feel that their primary use for their computer was education.

> ***"Yes, my wife thinks I am weird too."***

Such enjoyable learning can take many forms. It may be learning a new programming language, or using computer-aided instruction, or keeping up with developments in the computer field. Hobbyists enjoy learning in all these ways and more. Heath/Zenith computers are ideal for learning, especially when they've been built from a kit. After building or modifying their computer, hobbyists know what that little thingamajig next to the green doohickey really does. As George Galliher (Dalton, Massachusetts) put it, "I like to feel I can take the cover off and look inside and know what I am looking at."

The teaching role of a Heath/Zenith computer doesn't stop with its first owner, either. Emmet F. Freitas (San Jose, California), for instance, moved up to an H100 and sold his old H8 "at a giveaway price to a young fellow who will gain much more knowledge than he would have gained from a [new] computer that he would be able to obtain on his meager budget."

Although their main use of computers as a hobby is education, hobbyists also use their computers for a multitude of everyday tasks. As Bob Bortolin said about his computer, "It seems to be useful in just about everything I can imagine, whether it's planning a budget, balancing a checkbook, cataloguing video tapes, playing games, learning different programming languages or writing letters."

Ham radios, recipes, and other tasteful hobbies

Hobbyists also found their computers useful in conjunction with other hobbies, like operating ham radios, cataloguing collections, or keeping sports statistics. The most common use of the computer for these other hobbies was for amateur radio. Almost 20% of the hobbyists were ham radio operators.

These radio hams often use their computers as radio teletypewriters (RTTYs), which have a standard page width of 72 characters. Back in 1980, the H89—with its 80 columns and 25 lines—was the only available all-in-one micro that had a big enough screen to meet this requirement. That's the main reason why Ernest D. Kolb of New Providence, New Jersey, chose the H89 for use with his radio hobby.

Another ham operator who uses an H89 is Rick Swenton, from Bristol, Connecticut. He runs a VHF repeater facility for boosting the strength of radio signals, giving greater radio range to hams in the area. He is currently developing a microprocessor-based repeater controller. He uses the H89 to write the software and program the read-only memories for the controller. He can do all this on his inexpensive H89, whereas several years ago "you needed an Intel MDS development system costing plenty of bucks."

David C. Hood of Anderson, Indiana, writes that he too uses his H100 for an RTTY. He has found an excellent RTTY program that runs under the Zenith Disk Operating System and is in the public domain. But Mr. Hood's interest is really focused on packet radio. Packet radio is "the rapid (and error-free) communication between computers over radio and satellites.... It is like a modem over the air." He feels that its possibilities for future communication are unlimited.

Many hobbyists use their computers for cataloguing collections of one type or another. A few husbands mentioned cataloguing their wives' recipes—although the husband usually seemed more enthusiastic about the idea than the wife. Other respondents kept records of their music or videotape collections; and at least one, J. R. Bobbitt of Montclair, New Jersey, keeps track of Dungeons and Dragons characters on his H89.

Sports statistics also occupy an important place on the disks of many computers. Curtis Renwick of Indianapolis describes himself as a "numbers person." He keeps records of just about everything on his '89 including baseball, basketball, and football statistics and

predictions. And from Tallahassee, Florida, James K. Knerr writes that he develops and maintains his team's bowling scores on his H89. He is also in the process of developing another bowling system for Mixed League bowling.

The Computer Widow and other legends of the Information Age

People are finding an increasing number of uses for their computers as hobbies. But this heavy use of computers is not without its cost. Since the computer hobbyist usually seems to be the husband, the wife is often just an onlooker. So Mr. Manney's wife is not alone when she regards her husband as weird. Many other wives don't understand their husband's fascination with a computer, either. Ernest Kolb, for instance, hasn't been able to get his XYL (that's "wife" in ham-radio lingo) interested in computers at all. And Mr. Knerr's wife describes herself as a "computer widow."

On the other hand, some wives, like R. D. Hooper's, go against the odds and use the computer as avidly as their husbands. Mrs. Hooper is busy using their computer for her own hobby, creating crossword puzzles.

In a few cases, the rest of the family has also been demanding their share of computer time. The Hooper children use the family computers to develop and play games. David MacConnell's role has become that of a troubleshooter while his wife and 13-year-old son use the computer. Rick Swenton writes that his two-year-old daughter likes to play with the keyboard.

This early exposure to computers is important—these children are the first generation with the chance to grow up with a computer in the home. Such children represent the users of the future for the machine of the future, the computer.

But some people aren't content just to let the future happen—they are actively planning their future today. P. G. Manney, for example, is using his computer to design the engine of the future, one that will use any fuel and have only one moving part. He is convinced that its ruggedness will prove an asset in underdeveloped nations.

Walter M. Scott, III, uses his computer for what may be the home of the future. His H8, which he calls Dr. Bunsen Honeydew (after the Muppets character), controls his home. "Bunsen wakes me up, and controls lights and appliances, and reminds me of important events. He alerts me to visitors and mail, and answers the telephone." Mr. Scott communicates with Bunsen through wireless intercoms and touch-tone keypads throughout the house. He can even call Bunsen on the telephone and get a status report.

Computers are, without a doubt, the machine of the future—but they're also valuable for studying the past. Don Romie of Englewood, Ohio, plans to use his Z100 for his genealogy hobby. He will use the computer to store family information, print ancestral charts and, eventually, print a family history booklet.

The pros and cons of using Heath/Zenith computers

One of the reasons most often cited for using a Heath/Zenith computer for a hobby is the thorough and understandable documentation available. This is very important in the case of Richard C. Barnes. He lives on Mahe, an island in the Republic of Seychelles, in the middle of the Indian Ocean about 1,000 miles from the east coast of Africa. He has been working at a satellite-tracking station there since 1970. He feels that "if anyone can be called a self-made 'computerist' I can. Everything I learned, I learned from manuals, books, and just pure experimentation." That's because for two years he had the only computer in the entire country. (Talk about isolation!) It's in situations like this that one is really thankful for good documentation.

Just how good is this documentation of Heath/Zenith's that we've heard so much about? Rick Swenton said the following: "I am a field service tech and I feel that in some respects, Heath provided more hardware-related info than

my company provides me for my work."

The expandability of Heath/Zenith computers is another big plus for most hobbyists who are constantly wanting to add on to their machines. Bob Bortolin has an H8 and he feels that its design "allows easy upgrading, whether it's a higher density disk drive or a new central processing unit. It's usually just a simple matter of unplugging the old and replacing it with the new."

Heath hackers also have the advantage of the source code for the Heath Disk Operating System (HDOS)—which, according to J. R. Bobbitt, is "simple, direct, and complete." The availability of the HDOS source code was the biggest reason why Frank Davis of Green Bay, Wisconsin, bought a Heath computer to begin with.

Hobbyists feel that access to the HDOS source code helps them to really understand their computers, and they can then modify the HDOS operating system if they so choose. (See the article by Skip and Pam Chambers in *Sextant* #9, March/April 1984, "How One Family Learned to Live with an H89: A Dialogue," for an example of how one reader modified HDOS to fit his criteria.) At present, HDOS is the only operating system whose source code is readily available. Unfortunately, only the H8 and H89 computers have HDOS; the newer Heath/Zenith computers use other operating systems whose source code Zenith Data Systems can't make available since ZDS doesn't own Z-DOS or MS-DOS—Microsoft does.

Finally, the outstanding support for their computers from dealers, fellow owners, and Heath/Zenith has brought many sighs of relief from hobbyists with problems.

There are, of course, also some drawbacks to using computers for a hobby—which is probably why more people have a stamp collection rather than a computer for a hobby. Not only do computers cost considerably more than most hobbies, but there are a couple of problems which seem to be unique to Heath/Zenith computers.

For instance, there is the familiar problem of not enough advertising by Zenith. David Hood claims that "every time I see an ad on TV or in a magazine for other computers, I feel abandoned." Many Heath/Zenith owners complain that it's often hard to find the software that they need for their computer and they blame this on what they see as Zenith Data Systems' underexposure of their computers. As Jim Eilers phrased it, "There are some disadvantages to being the only Heath computer in town."

Because of the lack of software, Mr. Eilers feels like he has a "Mercedes-quality device and [I] am driving it in circles around and around a pasture in Oklahoma." To solve this software availability problem, a couple of readers suggested an S-100 card for the Z100 that would allow it to emulate the IBM Personal Computer's ROM, and thus run all the software that has been created for the PC. This would be very difficult, however. (Graham Wideman discussed some of the reasons why in his article in *Sextant* #9.)

A Dr. Bunsen in every home?

Today it seems as though the buzzword "computer literate" is on everyone's lips. If you aren't computer literate you won't get that job promotion. If your kids aren't computer literate they'll be behind everyone else in school, their career choices will be limited, they'll wind up working at a car wash the rest of their lives. So buy a home computer today. Buy an Atari, buy a Commodore, buy an Apple, and success is guaranteed.

Unfortunately, after all the hype, the home computer too often just languishes in a corner—its screen dark and silent. After all, many people wonder what more there is to home computers and computer literacy than the ability to play Pac-Man.

Plenty, according to our hobbyist respondents. Even if you've learned how to shoot down a thousand Xylons before being mortally wounded, you aren't necessarily computer literate. Computer literacy means being able to get a computer to do what you want it to do. It *doesn't* mean just doing what the computer wants you to do. Almost anyone can type a carriage return in response to a computer prompt. But, in order to get that box of whirring disks and blinking lights to do what you want it to do, you must understand it—how it doesn't work as well as how it does work.

It isn't hard to gain this knowledge. You can pore through thick manuals or attend summer computer camp; but those aren't fundamental to understanding. What is fundamental is the willingness to learn, a natural curiosity about what is really going on in there, behind that blinking screen. That's the first prerequisite for a computer hobbyist. (The second is access to a computer to put that curiosity to use!)

Is your computer sitting in the corner with a blank expression on its terminal? Well, it doesn't have to stay that way. You can use it to write that novel, control your ham radio station, or balance your checkbook. Or, if you prefer, you might use your computer the way Walter Scott did. Design your own "Bunsen" to wait on you so you don't have to get out of your easy chair. Either way, you're sure to learn something new and enjoy doing it. And that's really the key to becoming a computer hobbyist and making the best use of your Heath/Zenith computer.

Let Your '89 W.I.S.E. Up

Wisconsin Intelligent Systems Engineering's 2S+2P+RTC gives you two serial ports, two parallel ports, and a real time clock, all on one board.

James Bingham

If you have an H/Z89 computer, you probably got it with a serial input/output (I/O) board. It lets the '89 talk to serial printers quite nicely. It's also indispensable if you plan on using a modem, or if you're a hardware buff like me. Later on, you may think about picking up a parallel I/O to handle parallel printers (frequently faster) and graphic plotters. If you have a lot of peripherals connected to your computer and really want to save time, parallel ports are a necessity.

As for my computer, it didn't have either a serial or a parallel port. I picked up my '89 at a really good price, but it was bare bones. It did not take long to discover that I needed a printer—which put me in the market for an I/O board.

I considered the various options with an eye to the price-performance ratio. In doing so, I soon discovered the 2S+2P+RTC board offered by Wisconsin Intelligent Systems Engineering (W.I.S.E.). It was advertised as having two serial ports and two parallel ports in addition to a real-time clock.

I checked with a few members of the Phoenix Heath Users' Group, but could find no one with any knowledge of this board. Seeing as I was pretty well convinced that this was the best purchase for my money, I went ahead and ordered it, anyway. For a price of $155, I thought it was worth a try.

My first surprise came with the receipt of the board. I received it less than two weeks after placing my order. In this day and age, I find that really unusual. Shortly thereafter, I placed a second order, for the RS-232 serial cable. I also received that within two weeks. Immediately after that, I received an errata sheet with a few minor changes to the documentation that I had received with the board. All this fast service showed me that W.I.S.E. is concerned about customer satisfaction—which was a definite plus to start with.

As for the quality of the board, it is excellent—very well laid out and very sturdily constructed. The parts are all labelled. Overall, it appears to equal or exceed Heath's own standards. I definitely consider it an asset to my computer system.

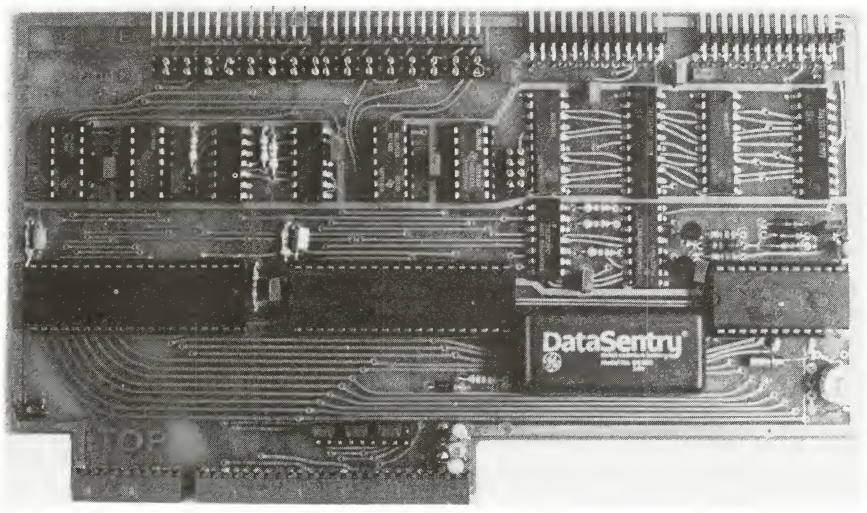


Photo courtesy of W.I.S.E.

Connecting the serial ports

The serial ports on this board are advertised to be compatible with the ports on the Heath serial board. The printer I added was a 99G from MPI, purchased at the local Heathkit Electronic Center. Ron, the manager there, told me that all I had to do was plug the cables into my '89 serial port and enjoy hard copy. Since I would be using the W.I.S.E. serial port instead of Heath's, this would be an excellent check of its compatibility.

After connection, however, nothing happened. I found that two of the handshaking signals were crossed between the printer and the 2S+2P+RTC board. (These signals are called RTS and DTR in the printer manual and DSR and CTS in the 2S+2P+RTC documentation.) I simply reversed these leads at the board's connector since this was the easiest place to do so. After this, the printer performed just fine.

Both of the 2S+2P+RTC serial ports are programmable for either modem or printer operation. The serial printer (LP) port comes already programmed for DCE operation, while the second serial port comes programmed for DTE operation.

The board's programmable jumpers are easy to work with. They are of the plug-in variety—which I thought was kind of nice. (The polarity of one of the handshaking signals may also be jumper selected.)

The parallel ports

The voltage levels of the parallel ports are compatible with digital-logic (TTL) chips. Cables are available from W.I.S.E. which interface these ports to a Centronics connector at the distant end.

Each of the parallel ports has two handshaking lines. However, all of these signals are available to the '89 only by accessing one of the two input ports. In essence, we can have either one parallel port with full handshaking or two free-running parallel ports (no handshaking).

I have connected my '89 to an Intel SDK-85 single-board computer using parallel I/O with full handshaking. I am able to operate full duplex very nicely. I am going to use this setup to do some experimental work on the SDK-85 since it has a wire-wrap area for breadboarding.

The real-time clock

The real-time clock (RTC) employs the MM58167 RTC chip which keeps time in increments down to .001 seconds. There is an optional battery pack available so that the clock will run even when the computer is turned off. The serial ports and the RTC may be programmed with plug-in jumpers to generate hardware interrupts; and it generates the same interrupt levels as Heath/Zenith products.

Software support

The documentation which comes with the board covers installation, operation, software, and schematics. Support software came both on a disk and in a printout. The disk format may be speci-

fied as either CP/M or the Heath Disk Operating System (HDOS).

The advertisement did not specify that software support was included in the purchase price; so I did not specify any disk format. I received the CP/M format, so it appears that this is the default disk you will receive if you don't specify. It worked out for me because I run both HDOS and CP/M.

The software includes support for serial I/O, parallel I/O, and the real-time clock. Most of the programs are written in assembly language, but there are a few written in BASIC. The documentation for the assembly-language programs is excellent. I had no trouble putting my board into operation.

For BASIC programmers, two programs are supplied with the W.I.S.E. board. The first program is for accessing the parallel ports. (You might use it with a joystick, for instance.) The second program is for reading the RTC. (The serial ports are compatible with Heath/Zenith products; so you can use any standard HDOS or CP/M software with them.)

I felt that a BASIC program for setting the RTC should also have been included, but it was not. For this reason, I wrote a complete program for the RTC. It is included here as Listing 1. This program is written in Microsoft BASIC Version 5.2 running under CP/M. But it should be adaptable to most BASICs.

Setting the RTC

The program shown in Listing 1 is pretty straightforward; only a few notes are called for.

Lines 80 through 370 acquire the time and date from the operator for setting the RTC. Lines 390 through 520 actually set the RTC.

The time and date are then read from the real-time clock and posted on the

Listing 1. A BASIC program to set the real-time clock of the 2S+2P+RTC board. The program is written in Microsoft BASIC Version 5.2 running under CP/M.

```
10 'INITIALIZE THE REAL TIME CLOCK.  SET TIME AND CHECK OPERATION
15 '                                   5 APRIL, 1983 BY JIM BINGHAM
20 '
30 'INITIALIZE VARIABLES
40 DIM D%(8): ON ERROR GOTO 1010
50 FOR I%=0 TO 8: D%(I%)=0: NEXT
60 '
70 'GET DATE
80 PRINT CHR$(27);"E";CHR$(27);"Y2 ";CHR$(27);"l";@
       "DATE FORMATTED AS DAY MONTH YEAR.  FOR EXAMPLE     2 JUNE 83":@
       PRINT CHR$(27);"Y$ ";CHR$(27);"l";: D%(6)=0: D%(7)=0: D%(8)=0
90 LINE INPUT "DATE PLEASE . . . ";A$: GOSUB 900
100 D%(6)=VAL(A$): IF D%(6)<1 OR D%(6)>31 THEN GOSUB 940: GOTO 80
110 A$=RIGHT$(A$,LEN(A$)-1): IF LEFT$(A$,1)<"A" THEN 110
120 B$=LEFT$(A$,3): RESTORE 970
130 FOR I%=1 TO 12: READ C$: IF C$=B$ THEN D%(7)=I%: I%=12
140 NEXT: IF D%(7)=0 THEN GOSUB 940: GOTO 80
150 A$=RIGHT$(A$,LEN(A$)-1): IF LEFT$(A$,1)<"0" OR LEFT$(A$,1)>"9" THEN 150
160 D%(8)=VAL(A$)
170 IF D%(8)>99 THEN D%(8)=D%(8)-100: GOTO 170
180 '
190 'GET DAY OF WEEK
200 PRINT CHR$(27);"Y% ";CHR$(27);:@
       LINE INPUT "lENTER DAY OF WEEK PLEASE (SAT, SUN,...,ETC.) . . .";A$:@
       GOSUB 900: D%(5)=0
210 B$=LEFT$(A$,3): RESTORE 980
220 FOR I%=1 TO 7: READ C$: IF C$=B$ THEN D%(5)=I%: I%=7
230 NEXT: IF D%(5)=0 THEN GOSUB 940: GOTO 200
240 '
250 'GET TIME
260 PRINT CHR$(27);"Y2 ";CHR$(27);"l";@
       "TIME FORMATTED AS HOUR MINUTE SECOND.  EXAMPLE 10:32:00 ":@
       PRINT CHR$(27);"Y& ";CHR$(27);"l";: D%(4)=0: D%(2)=0
270 LINE INPUT "ENTER TIME PLEASE . . . ";A$: IF LEFT$(A$,1)<"0" OR@
       LEFT$(A$,1)>"9" THEN GOSUB 940: GOTO 260                    →
```

terminal. This operation is performed by lines 540 through 860. The read portion of Listing 1 is a continuous loop and must be exited via a control-C. This read loop was incorporated to verify actual operation after setting.

If you wish to start the RTC at any seconds value other than zero, then you must delete lines 500 through 520.

Note that the error traps (lines 1000 through 1070) are very general purpose in nature. If you key in this program, therefore, I recommend that you exclude these lines until the remainder of the program has been run and checked for syntax errors.

One other point concerns the read routine. Should you decide to extract it for use in another program, be sure to prefix it with lines 390 and 400. This is necessary because the read routine is worthless unless the RTC has previously been selected by lines 390 and 400.

To summarize, I can say that both the 2S+2P+RTC board and the documentation are excellent. At the price and for this versatility, I do not believe you can beat it.

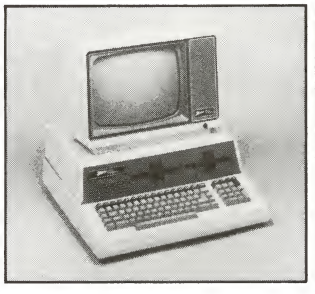
I was very pleased with the service and like the availability of options such as the battery pack and the cables. Should this article influence you to purchase this board, I do not believe that you will be disappointed.

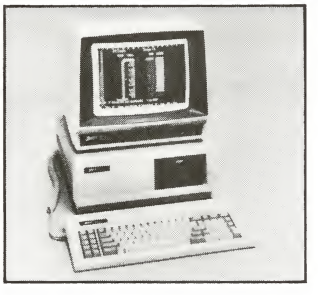
```
280 D%(4)=VAL(A$): IF D%(4)>24 THEN GOSUB 940: GOTO 260
290 A$=RIGHT$(A$,LEN(A$)-1): IF LEFT$(A$,1)<"0" OR LEFT$(A$,1)>"9" THEN@
        A$=RIGHT$(A$,LEN(A$)-1) ELSE 290
300 IF LEFT$(A$,1)<"0" OR LEFT$(A$,1)>"9" THEN GOSUB 940: GOTO 260
310 D%(3)=VAL(A$): IF D%(3)>60 THEN GOSUB 940: GOTO 260
320 A$=RIGHT$(A$,LEN(A$)-1): IF LEFT$(A$,1)<"0" OR LEFT$(A$,1)>"9" THEN@
        A$=RIGHT$(A$,LEN(A$)-1) ELSE 320
330 IF LEFT$(A$,1)<"0" OR LEFT$(A$,1)>"9" THEN GOSUB 940: GOTO 260
340 D%(2)=VAL(A$): IF D%(2)>60 THEN GOSUB 940: GOTO 260
350 '
360 'CONVERT VARIABLES TO PACKED BCD FORMAT
370 FOR I%=2 TO 7: X%=D%(I%): Z%=FIX(X%/10): X%=X%-Z%*10: Z%=Z%*16:@
        D%(I%)=X% OR Z%: NEXT
380 '
390 'SELECT REAL TIME CLOCK
400 X=(INP(&HE4) AND &HF3)+4: OUT &HE4,X
410 '
420 'RESET COUNTERS
430 X=(INP(&HDC) AND &HF3)+4: OUT &HDC,X
440 OUT &HD2,&HFF
450 '
460 'SET TIME
470 X=(INP(&HDC) AND &HF3)+&HC: OUT &HDC,X: Y=&HD0
480 FOR I%=0 TO 7: OUT (Y+I%),D%(I%): NEXT
490 '
500 'SYNCHRONIZE THE CLOCK
510 X=(INP(&HDC) AND &HF3)+4: OUT &HDC,X
520 OUT &HD5,&HFF
530 '
540 'READ REAL TIME
550 '
560 'INITIALIZE THE VARIABLES
570 FOR I%=0 TO 7: D%(I%)=0: NEXT: S%=0
580 '
590 'READ COUNTERS
600 X=(INP(&HDC) AND &HF3)+&HC: OUT &HDC,X
610 FOR I%=0 TO 7: D%(I%)=INP(&HD0+I%): NEXT
620 '
630 'UNPACK DIGITS
640 D%(0)=(D%(0) AND &HF0)/16            'DECODE THOUSANTHS
650 D%(0)=D%(0)+(D%(1) AND &HF0)*10      'DECODE HUNDRETHS
660 D%(0)=D%(0)+(D%(1) AND &HF)*100      'DECODE TENTHS
670 FOR I%=2 TO 7: Z%=((D%(I%) AND &HF0)/16)*10: D%(I%)=(D%(I%) AND &HF)+Z%:@
        NEXT
680 D1=D%(2)+D%(0)/1000
690 '
700 'HAS THE SECOND'S DIGIT CHANGED?
710 IF D%(2)=S% THEN 600 ELSE S%=D%(2)
720 '
730 'IF SO, READ IT OUT
740 'BEGIN WITH DAY OF WEEK
750 RESTORE 980
760 FOR I%=1 TO D%(5): READ A$: NEXT
770 PRINT A$;" ";
780 '
790 'NOW DO DATE
800 PRINT D%(6);" ";: RESTORE 970
810 FOR I%=1 TO D%(7): READ A$: NEXT
820 PRINT A$;" ";D%(8);"     ";
830 '
840 'FINALLY THE TIME OF DAY
850 PRINT D%(4);":";D%(3);":";D1
860 GOTO 600
870 '
880 '
890 'MAP TO UPPER CASE
900 FOR I%=1 TO LEN(A$): B$=MID$(A$,I%,1):@
        IF B$>"Z" THEN IF B$<"{" THEN MID$(A$,I%,1)=CHR$(ASC(B$)-32)
910 NEXT: RETURN
920 '
930 'ERROR FLAG
940 PRINT CHR$(7);CHR$(27);"Y5 INVALID ENTRY. HIT RETURN TO CONTINUE";
950 LINE INPUT A$: RETURN
960 '
970 DATA JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
980 DATA SUN,MON,TUE,WED,THU,FRI,SAT
990 '
1000 'ERROR TRAPS FOLLOW
1010 G%=0
1020 IF ERL<350 THEN G%=3
1030 IF ERL<240 THEN G%=2
1040 IF ERL<180 THEN G%=1
1050 IF G% THEN RESUME 1070
1060 ON ERROR GOTO 0
1070 GOSUB 940: ON G% GOTO 80,200,260
```

Ordering Information

2S+2P+RTC, $155.
Without UARTs and RS-232 ICs, for customers with existing serial boards, $110.
Optional rechargeable battery, $10.
Foreign orders add $15 for shipping and handling.
Wisconsin Intelligent Systems Engineering
Box 344, 422 Third Street
Baraboo, WI 53913
608/356-9432

Finally, text processing that fits your computer to the letter.





Newline. Matched to the Zenith 100, 150 and the IBM-PC. If you have one of these systems, Newline has the right line of text processing software for you. Because Newline's Professional Text Processor (PTP) is matched to the keyboard and display characteristics of each of these computers.

There are no complicated key sequences to learn. And you'll be able to use labeled editing keys. Which means it's exceptionally easy to use.

New features for Newline's PTP. Even better, now PTP has more powerful text processing than ever before. There's everything from full screen text editing and on-screen bold, underline, paragraph fill and justification to cut and paste to configurable macro keys and much more. Plus, you'll be able to use our software with any printer.

Update from our old line. If you're presently using the Newline TxtPro software on your Zenith 100, now you can update to our more powerful version. It's called the PTP-100. If you currently have the ZDOS TxtPro, you can upgrade to ZDOS PTP-100. If you have the CP/M-85 TxtPro, you can upgrade to CP/M-86 PTP-100, but you'll also need to upgrade your system to CP/M-86. Just specify which one you have when you order. And if you return your old TxtPro disk to us with your order, you qualify for a special reduced price.

For Zenith 150 and IBM-PC users, there's the new PTP-PC. And it has all the same features as the PTP-100.

Software development editing. For programmers, Newline's PTP also offers auto indent and produces ASCII files for use with assemblers, interpreters and compilers. And that's not just different, it's unique.

Just give us the word. Newline's Professional Text Processor (PTP) is available right now. So place your order today. And get the text processing software that fits *your* system to the letter.

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

PTP-100 (for Z100) @ **$99. or**
PTP-100 (Z100 Upgrade) @ **$50.***

	Qty	Amount
☐ CP/M-86 (replaces CP/M-85)	____	____
☐ ZDOS	____	____

*To qualify for Z100 upgrade price of $50. you must return your TxtPro disk.

PTP-PC (for Z150 or IBM-PC) @ **Special Introductory Price of $149.**

☐ CP/M-86	____	____
☐ MS-DOS or PC-DOS	____	____
RI Residents Add 6% Sales Tax		____
Shipping ($3. per program)		____
TOTAL ENCLOSED		____

Payment in U.S. Funds Only • Allow 2 wks. for Personal Checks • CALL (401) 624-3322 FOR C.O.D. DELIVERIES

NEWLINE SOFTWARE

P.O. Box 289 • Tiverton, RI 02878 • (401) 624-3322

Circle #145 on Reader Service Card

File Management Under Lucidata Pascal

These extensions to Pascal make input and output vastly easier.

Donald E. Risher

Over the past couple of years, I have read and enjoyed a number of articles relating to the Lucidata Pascal package available from Polybytes of Cedar Rapids, Iowa. However, I have yet to see an extensive discussion on some of its more technical aspects.

To rectify that somewhat, this article will briefly discuss file management and some of the extensions to standard Pascal that assist in the creation and disposition of files within the Polybytes system. I will also cover some other powerful extensions useful to a computer "hacker" like myself.

In case you want some background on the Polybytes implementation of Lucidata Pascal, you might look at Hugh Kenner's article in *Sextant* #6, Summer 1983 ("Pascal Elucidated"). Also, Henry Fale has a series of articles in *REMark*, the national Heath Users' Group publication. (See "Pascal Corner," in *REMark*: issue #22, November 1981, p. 1; #23, December 1981, p. 24; #25, February 1982, p. 6; #26, March 1983, p. 6; and #29, June 1983, p. 32.)

For more background on Pascal, using Tiny Pascal as an example, William Tavolga has a three-part "Introduction to Pascal" in *Sextant* #11, July-August 1984, and #12 and #13. It is aimed at the BASIC programmer moving up to Pascal. (Most articles on Pascal, including this one, are written on the assumption that you will already have gotten your feet wet in programming before coming to Pascal.)

The Polybytes Lucidata Pascal package is available in versions for both the Heath Disk Operating System (HDOS) and CP/M. Whichever you choose, the compiler can take the program you write and produce a final version in either of two forms: a .BIN file; or an .ABS (HDOS) or .COM (CP/M) file. The .ABS or .COM file, of course, can run on its own. The .BIN file will require Pascal's runtime system (PRUN) to run "alongside" it, to handle things such as feeding your program math routines, etc.

For opening and operating on disk data files, the Polybytes compiler has four operations that will interest us here: ASSIGN, RESET, REWRITE, and POSITION.

To ease you through the manipulation of these files, Lucidata has what are referred to as "pragmats," indicated by the curly-bracket dollar-sign symbol ({$). These are extensions to the Pascal language designed to eliminate some of the input/output (I/O) weaknesses of the standard language.

Setting up file buffers

First, a word about file buffers. The runtime system, PRUN, communicates with the disk operating system by means of a maximum of six blocks of memory which handle file management. Each of these file-control blocks is 285 bytes in size. There is also a link channel reserved for internal management.

A file-control block is set aside in memory for each file you designate in the command line.

What this means to the average "home-computer type" programmer is that at any given time during the execution of a program, six files may be opened for data input and output, whether it be to a listing device or disk file.

During the loading of a program into random-access memory (RAM), the runtime system scans the PRUN command line for your input invoking your program from the terminal. File buffers are allocated accordingly—that is, a file-control block is set aside in memory for each file you designate in the command line.

This allocation of file buffers is directly related to the file declarations in the main program block. (You don't have to declare your files on the command line, but we'll get into that later.)

Example 1 is based on my initial version of the CHECKING program found in Listing 1. It gives the command line for invoking the program; and it shows the first five lines of the program, with the program header as the first line.

In Example 1, DEV: is the reference to any of the mounted disk drives in the system; PRUN is the call to the Polybytes runtime system; CHECKING is the program name; and LP: the listing device. CHECFILE and ACCNTBAL are data files. This establishes three control blocks, or buffers—for LP:, CHECFILE, and ACCNTBAL.

When you declare files on the command line, it is very important that those data files or listing devices correspond to the files declared in the main program header. They must also correspond to the declaration of files in the body of the source code.

In Example 1, you can see the relation to each other of the runtime command line, the program header statement, and the file declarations. Note that LP: is listed first in the command line invoking the program. PRINTER is listed first in the main procedure header (line 1). And PRINTER : FILE OF CHAR; (line 3) is listed as the first variable. DEV:CHECFILE, CHECFILE, and CHECFILE : FILE OF CHECKS are listed second; and DEV: ACCNTBAL, ACCNTBAL, and ACCNTBAL : FILE OF CHAR third.

This convention, linking file specifiers in the command line, the program header, and file declarations in the main program block, must be followed exactly. Lucidata expects the order followed

```
      DEV:PRUN DEV:CHECKING LP: DEV:CHECFILE DEV:ACCNTBAL

1     PROGRAM CHECKING(PRINTER, CHECFILE, ACCNTBAL);
2     VAR
3        PRINTER   : FILE OF CHAR;
4        CHECFILE  : FILE OF CHECKS;     (* RECORD TYPE *)
5        ACCNTBAL  : FILE OF CHAR;
```

Example 1. The initial version of the command line and the first five lines of the program CHECKING.PAS (Listing 1).

in the PRUN command line to be followed in the program headers and file declarations.

During the execution of the program, the creating and accessing of files do not have to follow the order of declaration in the program. But the file declarations (as in Example 1, lines 3 through 5) must correspond to the command line specifiers and to the program header (line 1). Otherwise, data will more than likely end up in left field.

One must also be aware that a file is declared by type—CHAR, BYTE, RECORD, etc. It is impossible to access a file of CHAR to fill a variable of another type such as RECORD. There are exceptions to this rule, but we will pass them by at this point.

ASSIGN

Of the four operations for handling disk data files, we'll begin with the procedure ASSIGN. You will be very restricted in your applications of data files if you neglect this important aspect of the compiler—as I did initially.

When I wrote the original version of my checking program, it was my first major effort in Pascal. I was not at all familiar with all of the extensions—the key built-in compiler procedures and functions supported by Polybytes—most notably, ASSIGN.

The important point about the ASSIGN procedure is that it eliminates the necessity of declaring file names in the PRUN command line that calls the program. Using Lucidata Pascal without ASSIGN could lead to some insurmountable problems when you consider the limited keyboard-input buffer of the H8 and H/Z89 (or for that matter, any of the currently popular microcomputers).

For example, a bowling program I wrote in Benton Harbor BASIC maintains weekly and seasonal statistics for up to 24 teams in a league. This program

Listing 1. Excerpts from CHECKING.PAS, a personal checking program. Only those procedures applicable to the discussion are included.

```
1     PROGRAM CHECKING (AFILE, BFILE);
2     (* Runtime command line DEV:PRUN DEV:CHECKING *)
3     CONST
4         LENGTH = 30;
5         PAGE   = CHR(12);       (* Places cursor mid-page *)
6     TYPE
7         STRING = ARRAY [ 1 .. LENGTH ] OF CHAR;
8         LINE   = ARRAY [ 1 .. 9 ] OF CHAR;
9         CHECKS = RECORD
10                   CKNO    : INTEGER;
11                   PAYEE   : STRING;
12                   AMOUNT  : REAL;
13                   DATE    : LINE;
14                   CKOFF   : BOOLEAN
15                   END;
16    VAR
17        CHECK    : CHECKS;
18        FINISHED : BOOLEAN;
19        BALANCE  : REAL;
20        LATEST, SELECTION, CHECKNO : INTEGER;
21        AFILE    : FILE OF CHECKS;
22        BFILE    : FILE OF CHAR;
23
24    (*$A = $22DF*)
25    BUFFMODE : BOOLEAN;         (* Buffered input *)
26    (*$A = $20BF*)
27    DATESTRING : LINE;          (* System date *)
28    (*$A = $22E0*)
29   DERRPR : BOOLEAN;            (* Reset error *)
30    (*$S GO BACK TO STACK *)
31
32    PROCEDURE CLEARSCREEN;      (* Clear screen *)

38
39    PROCEDURE PRINTER_READY;    (* Check printer status *)
40    CONST
41       PORTNUMBER = 208; ESC = CHR(27);
42    VAR
43       STATUS, READY : INTEGER;
44
45       FUNCTION INPORT(PORTNUMBER : INTEGER) : INTEGER;
46       EXTERNAL $22E4;
47
48       PROCEDURE TWENTY_FIFTH_LINE (MODE : INTEGER);
49
50       (* Procedure to enable 25th line, direct the cursor and *)
51       (* reverse video *)
52
53       CONST
54          V = CHR(56); H = CHR(32);      (* Direct cursor address *)
55       BEGIN
56          CASE MODE OF
57             0 : WRITE(ESC,'o',ESC,'y1',ESC,'k',ESC,'q');
58             (* Disable 25th line *)
59             1 : WRITE(ESC,'j',ESC,'x1',esc,'Y',V,H,ESC,'p');
60             (* Enable 25th line  *)
61          END (* CASE *)
62       END; (* 25th line  *)
63
64       PROCEDURE PORTSTATUS;
65       (* Get status byte and unpack *)
66       BEGIN
67          STATUS := INPORT(PORTNUMBER);
68          READY := UNPACK(STATE,7,1)
69       END; (* Portstatus *)
70
71    BEGIN  (* Printer_Ready *)
72       PORTSTATUS;
73       IF READY = 1 THEN
74          BEGIN
75             TWENTY_FIFTH_LINE(READY);
76             WRITE(' Printer is OFF...or printer is OFF LINE!!!! ');
77             REPEAT
78                PORTSTATUS;
79             UNTIL READY = 0;
80             TWENTY_FIFTH_LINE(READY)
81          END (* IF *)
82    END; (* Printer_Ready *)
83
84    PROCEDURE STARTING_BALANCE;       (* Input balance from file *)
85    BEGIN
86      DERRPR := FALSE;                (* Disable returned error *)
87      ASSIGN(BFILE,'ACCNTBAL');       (* Link ACCNTBAL to BFILE *)
88      RESET(BFILE);
89      IF NOT EOF(BFILE) THEN          (* Is file on disk *)
90        BEGIN                         (* Yes, then read it *)
```

→

```
91          READ(BFILE,BALANCE,CHECKNO,LATEST);
92          RESET(BFILE);
93          DERRPR := TRUE
94        END
95      ELSE
96        BEGIN                          (* No, get first check no. *)
97          DERRPR := TRUE;
98          CLEARSCREEN; WRITELN(PAGE);
99          BALANCE := 0.0;
100         WRITE('What Check number do you wish to start account with? ');
101         READLN(CHECKNO); LATEST := CHECKNO
102       END
103   END;
104
105   PROCEDURE GET_BALANCE;  (* Show balance in account *)

116
117   PROCEDURE DEPOSIT;      (* Make a deposit *)

127
128   PROCEDURE MENU;         (* Show menu *)

146
147   PROCEDURE VOID_CHECK;   (* Void a check *)

172
173   PROCEDURE PRINT_CHECK;  (* Print and file check *)
174   VAR
175     CH : CHAR;
176     I  : INTEGER;
177   BEGIN
178     CLEARSCREEN;
179     WRITELN(PAGE);
180     PRINTER_READY;            (* Check printer status *)
181     ASSIGN(BFILE,'LP:');      (* Link LP: to BFILE *)
182     REWRITE(BFILE);           (* Open LP: *)
183     CHECK.CKNO := CHECKNO;
184     WRITE('  Are checks in printer and alinged? ');
185     READLN(CH);
186     WRITELN(BFILE);
187     WRITELN(BFILE);
188     ASSIGN(AFILE,'CHECFILE');(* Link CHECFILE to AFILE *)
189     REWRITE(AFILE);           (* Open CHECFILE *)
190     REPEAT
191       CLEARSCREEN;
192       WRITELN(PAGE);
193       WRITE('   Enter payees name: ');
194       READLN(CHECK.PAYEE);
195       WRITE('   Enter amount: ');
196       READLN(CHECK.AMOUNT);
197       BALANCE := BALANCE - CHECK.AMOUNT;
198       CHECK.CKOFF := FALSE;
199       IF NOT EOF(AFILE) THEN
200         BEGIN
201           POSITION(AFILE,CHECK.CKNO-LATEST); (* Position file *)
202           WRITE(AFILE.CHECK)                 (* Write it *)
203         END
204       ELSE WRITE(AFILE.CHECK);               (* Its first check *)
205       WRITE(BFILE,CHECK.DATE:11,CHECK.CKNO:10,CHECK.PAYEE:39);
206       WRITELN(BFILE,CHECK.AMOUNT:6:2,BALANCE:10:2);
207       FOR I := 1 TO 13 DO WRITELN(BFILE);
208       WRITE(BFILE,CHECK.PAYEE:36,CHECK.DATE:12,CHECK.CKNO:9);
209       WRITE(BFILE,'           ');
210       WRITE(BFILE,'$***');
211       IF CHECK.AMOUNT <10.0 THEN WRITELN(BFILE,CHECK.AMOUNT:4:2)
212       ELSE IF (CHECK.AMOUNT > 9.99) AND (CHECK.AMOUNT <100.0) THEN
213       WRITELN(BFILE,CHECK.AMOUNT:5:2)
214       ELSE WRITELN(BFILE,CHECK.AMOUNT:6:2);
215       FOR I := 1 TO 27 DO WRITELN(BFILE);
216       CHECK.CKNO := CHECK.CKNO +1;
217       WRITELN;
218       WRITELN;
219       REPEAT
220         WRITE('  Any more checks to write? <Y/N> ');
221         READLN(CH);
222       UNTIL CH IN ['Y','y','N','n'];
223     UNTIL CH IN ['N','n'];
224     REWRITE(AFILE);                    (* Close CHECFILE *)
225     REWRITE(BFILE);                    (* Close LP: *)
226     CHECKNO := CHECK.CKNO
227   END;
228
229   PROCEDURE LIST_CHECKS; (* List all or part of checks in active file *)

261
262   PROCEDURE CHECK_STATEMENT; (* Verify bank statement *)

313
314   BEGIN    (* MAIN PROGRAM *)
```

→

could conceivably maintain as many as 124 individual and team file names, plus half as many more for substitute bowlers. There is absolutely no way that the PRUN command line for a similar Pascal program could input that many individual file names. ASSIGN would be imperative.

If, in Example 1, I had used ASSIGN, that would have considerably simplified the runtime command line to invoke the program, and also the program file declarations. In the process, I would have eliminated in large part the problem of exact ordering of file specifications in the command line.

The CHECKING program shown in Listing 1 is the final version. (As presented here, it is condensed to show only those procedures pertinent to this discussion.) As indicated in the comment in line 2 of Listing 1, utilizing the ASSIGN procedure allows the command line to become:

```
DEV:PRUN DEV:CHECKING
```

instead of the longer form used in Example 1.

The main procedure block heading, or program header (Listing 1, line 1) becomes:

```
PROGRAM CHECKING (AFILE, BFILE);
```

And in the main program block, the file declarations become as shown in lines 21 and 22:

```
VAR
. . .
    AFILE : FILE OF CHECKS;
    BFILE : FILE OF CHAR;
```

With the use of the ASSIGN procedure, the data files ACCNTBAL (FILE OF CHAR) and CHECFILE (FILE OF CHECKS), and the listing device LP: (FILE OF CHAR), can be "assigned," or linked, to the files in lines 21 and 22 (AFILE and BFILE) during program execution.

One important point about file-buffer allocation. Suppose that, during the development of this program, I wished to make a listing on the printer of all the

```
315     BUFFMODE := TRUE;
316     CHECK.DATE := DATESTRING;
317     STARTING_BALANCE;
318     FINISHED := FALSE;
319     REPEAT
320       MENU;
321       CASE SELECTION OF
322           1  : DEPOSIT;
323           2  : GET_BALANCE;
324           3  : PRINT_CHECK;
325           4  : VOID_CHECK;
326           5  : CHECK_STATEMENT;
327           6  : LIST_CHECKS;
328           7  : FINISHED := TRUE
329       END;  (* CASE *)
330     UNTIL FINISHED;
331     ASSIGN(BFILE,'ACCNTBAL');
332     REWRITE(BFILE);
333     WRITE(BFILE,BALANCE,CHECKNO,LATEST);
334     REWRITE(BFILE);
335     BUFFMODE := FALSE
336   END.  (* checking *)
```

checks in the active check file—while still having access to the file ACCNTBAL. This implies that three channels, or buffers, would be open at the same time. And I have only two buffers available (AFILE and BFILE).

So the command line would have to reflect this additional file. And I would have to declare another file, CFILE: FILE OF CHAR, in the main body of the program; and CFILE: in the program header: PROGRAM CHECKING (AFILE, BFILE, CFILE); .

Using $F on the command line

There is one other consideration, the PRUN command line to call the program.

As noted above, you don't have to specify all your files on the command line. The runtime system is configured such that it automatically allocates two file buffers for access to files through the ASSIGN procedure. In the example above, two files were declared, AFILE and BFILE. No special consideration was given to the allocation of additional file buffers in the runtime command line. If CFILE is declared, then the command line must reflect this additional file buffer.

We can allow for this additional file buffer with a "pragmat." In this specific instance, that is the pragmat $F. (We'll discuss pragmats a little more thoroughly later on.)

To accommodate three simultaneous file accesses with the ASSIGN procedure, the following modification to the command line must be made:

DEV:PRUN DEV:CHECKING $F:3

Here, the pragmat $F:3 reserves three file buffers for program file access. (This is very similar to file reservations in Microsoft BASIC.)

Actually, pragmat $F:n can reserve any number of file buffers, up to the maximum of six mentioned earlier. The runtime system reserves two automatically. So the $F:n pragmat would be necessary only when more than two buffers are needed at any given time during the execution of the program. $F:3 reserves three buffers in this case, which includes the two that would be reserved anyway.

If you were to specify two files in the PRUN command line and also wished to reserve two file buffers for use with the ASSIGN procedure, then the total number of buffers required would be four. The pragmat $F:4 would be necessary in the command line, as in:

DEV:PRUN DEV:PROGRAM LP: DEV:FNAME $F:4

Here, DEV:PRUN calls the runtime system; DEV:PROGRAM specifies the program name; LP: is the listing device; DEV:FNAME lists a specific file on disk to be used in the program; and finally $F:4 tells the runtime system that two additional file-control blocks will be required to support unspecified files through the ASSIGN procedure during the run of the program.

Using ASSIGN in the body of the program

Now let's see how the ASSIGN procedure works in the body of the checking program in Listing 1. Using the example above, we will open data file "ACCNTBAL," as in procedure STARTING_BALANCE (lines 84-103). This file contains the balance in the account (BALANCE), the number of the next check to be written (CHECKNO), and the number of the first check in the active check file (LATEST).

Without ASSIGN, the following TYPE and VAR declarations would have to be made in the body of the main procedure:

```
TYPE
   AFNAME = ARRAY [1..8] OF CHAR;
VAR
   FNAME : AFNAME;
```

Then in the procedure STARTING_BALANCE, we could place

FNAME := 'ACCNTBAL';

This would assign ACCNTBAL to the variable FNAME. Then

ASSIGN(BFILE,FNAME);

would link FNAME to BFILE.

All this is very "tidy" and proper programming practice. But it could just as properly be boiled down to the simple statement in line 87:

ASSIGN(BFILE,'ACCNTBAL');

Here, we use the actual name of the file in single quotes—without the bother of declaring a type AFNAME and variable FNAME.

The runtime system links ACCNTBAL with the second file buffer BFILE to open a FILE OF CHAR for data input/output. To open and read (as in line 88 of the procedure), we would

RESET(BFILE);

That would open the file for a read operation. Then line 91

READ(BFILE,BALANCE,CHECKNO,LATEST);

reads the data into the variables on the stack.

Reader Service Card
January-February 1985
Expires April 15, 1985

Name _____

Address _____

_____

City _____ **State** _____ **Zip** _____

For further information from specific advertisers, refer to index on page 82 and circle numbers below.

100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249

Please check *only* one box for each of the following questions.

Which microcomputer system do you own/use?
A. ☐ H88 E. ☐ H8 I. ☐ H/Z150
B. ☐ H89 F. ☐ H11 J. ☐ H/Z160
C. ☐ Z89 G. ☐ H100/110/120
D. ☐ Z90 H. ☐ Z100/110/120

What is your principal occupation?
K. ☐ Engineer/Scientist P. ☐ Technician
L. ☐ Business Owner/Manager Q. ☐ Professional
M. ☐ Military R. ☐ Student
N. ☐ Programmer/Analyst S. ☐ Government
O. ☐ Educator T. ☐ Other

How did you obtain this copy of *Sextant*?
8. ☐ Subscription 10. ☐ Friend 13. ☐ Other
9. ☐ Retail Store 12. ☐ Single Copy Mail Order

Please rate the articles in this issue for interest and helpfulness using the key on page 83.

1	2	3	4	5	6	7	8	9	10	11	12	13	
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Excellent
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Good
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Fair
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Poor
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Didn't Read

Comments: _____

_____

_____

82a1

Reader Service Card
January-February 1985
Expires April 15, 1985

Name _____

Address _____

_____

City _____ **State** _____ **Zip** _____

For further information from specific advertisers, refer to index on page 82 and circle numbers below.

100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249

Please check *only* one box for each of the following questions.

Which microcomputer system do you own/use?
A. ☐ H88 E. ☐ H8 I. ☐ H/Z150
B. ☐ H89 F. ☐ H11 J. ☐ H/Z160
C. ☐ Z89 G. ☐ H100/110/120
D. ☐ Z90 H. ☐ Z100/110/120

What is your principal occupation?
K. ☐ Engineer/Scientist P. ☐ Technician
L. ☐ Business Owner/Manager Q. ☐ Professional
M. ☐ Military R. ☐ Student
N. ☐ Programmer/Analyst S. ☐ Government
O. ☐ Educator T. ☐ Other

How did you obtain this copy of *Sextant*?
8. ☐ Subscription 10. ☐ Friend 13. ☐ Other
9. ☐ Retail Store 12. ☐ Single Copy Mail Order

Please rate the articles in this issue for interest and helpfulness using the key on page 83.

1	2	3	4	5	6	7	8	9	10	11	12	13	
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Excellent
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Good
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Fair
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Poor
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Didn't Read

Comments: _____

_____

_____

82a2

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 27346 PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

Sextant Reader Service
P.O. Box 13558
Philadelphia, PA 19101

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 27346 PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

Sextant Reader Service
P.O. Box 13558
Philadelphia, PA 19101

ANNOUNCING THE MODIFIER

A disk utility that modifies the CP/M BIOS to be able to read and write to a number of 5.25" CP/M disk types.

There is a growing need for the everyday user of computer systems to be able to take data files home from the office to continue to work on them. The computers at home and at work may both run a version of CP/M, but the disk structures may be incompatible. This is especially a problem in the 5.25" world. MODIFY 89 was designed to address this problem. MODIFY 89 makes the CP/M operating system access a specified 5.25" drive as one of the below disk types.
Disks placed in that drive that are of the specified type can be used as if they were one of the standard disk types accepted by the H8, H/Z89 or H/Z90 computers. Thus PIP, STAT, DIR and others will work for that disk also. The price for MODIFY 89 is $49.95.

MODIFY 89 is set for the following disk types:

- Access S.S. D.D.
- Cromemco S.S. D.D.
- DEC VT180 S.S. D.D.
- IBM PC/Zenith 100 (CP/M) S.S. D.D.
- IBM PC/Zenith 100 (CP/M) D.S. D.D.*
- Kaypro II S.S. D.D.
- Morrow Micro Decisions S.S. D.D.
- NEC PC-8001A S.S. D.D.
- Osborne S.S. S.D.
- Osborne S.S. D.D.
- Otrona D.S. D.D.*
- Superbrain Jr. S.S. D.D.
- TI Professional S.S. D.D.
- TRS-80 Model I (Omnikron CP/M)
- TRS-80 Model III (MM CP/M)
- Xerox 820 S.S. S.D.
- Xerox 820-II S.S. D.D.
- Standard (Tests for H/Z37 and C.D.R. Disk types)

* = Double sided 5.25" drive required

S.S. = single sided, D.S. = double sided, S.D. = single density, D.D. = double density

Limitations: MODIFY 89 is not a disk duplicate program. It is currently available for use with an H/Z89 or H/Z90 computer that has an FDC-880H double density 8" and 5.25" controller, using C.D.R.'s BIOS V.2.9 or with an H8 computer using the FDC-H8 by C.D.R. Systems, Inc.

MODIFY 100 will soon be released for the Z100 line of computers at a price of $75.00.

Contact:

C.D.R. Systems, Inc.
7210 Clairemont Mesa Blvd., San Diego CA 92111
Telephone: (619) 560-1272
Or a C.D.R. Systems, Inc. dealer near you.

Circle #108 on Reader Service Card

INTELLIBURNER

EPROM–EEPROM–MICROCOMPUTER PROGRAMMER

UNIVERSAL PROGRAMMING CAPABILITIES AT AN AFFORDABLE PRICE

- **Ultra Fast Programming – 2716's in 16 Seconds**
- **Programs & Verifies 8K thru 256K Single Voltage EPROMs**
- **Erases, Programs & Verifies 2815 & 2816 EEPROMs**
- **Programs & Verifies 8748 and 8751 Series MICROCOMPUTERS***
- **Programming Chracteristics Selected by Convenient Personality Jumper Plug (DIP Header)**
- **Program, Verify, Status, & Diagnostic Display with Tricolor LED**
- **Serial Interface — 3, 4, or 5 wire — 1200 to 19200 Baud**
- **Supports XON/XOFF and READY/BUSY Protocols**

NO SPECIAL SOFTWARE REQUIRED. Transfer disk files (Intel Hex Format) to EPROM with your system's line printer or modem software. Transfer EPROM contents to disk file in Intel Hex Format with your system's modem software. Or use the supplied software ** to transfer any binary or ASCII file to/from EPROM.

PROGRAMS:

2758	2716	27128	27C16	8741*	8748*
2516	2732	27128A	2815	8742*	8749*
2532	2732A	27256	2816		8751*
2564	2764	68764	2817	X2212*	8755*

*Requires low cost personality adapter

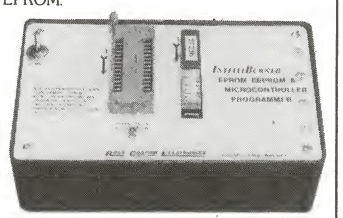
SOFTWARE AVAILABLE FOR: **

CP/M systems on 8" SSSD - many 5¼ formats
TRS-80 Model I & III TRSDOS
Heath H8/H89 HDOS & CP/M
ZENITH Z90 & Z100 CP/M - Z100 ZDOS
KAYPRO II/IV
IBM PC-DOS - many MS-DOS systems

IntelliBurner Programmer with Software	$ 299.00
RS-232 Interconnect Cable	12.00
Intelliburner PC Boards, EPROM, Plans & Software	99.00

Low Cost "DumBurner" serial programmers harness the power of your personal computer with the supplied software for full programming capabilities:

DumBurner II Programmer for 28 Pin and 24 Pin EPROMs and EEPROMs (with Software)	$ 199
DumBurner II Bare PC Board, Plans & Software	59
16K/32K DumBurner for 24 Pin EPROMs (with Software)	149
16K/32K DumBurner PC Board, Plans & Software	39



ROSS CUSTOM ELECTRONICS
1307 Darlene Way - Suite A12
Boulder City, Nevada 89005
PHONE (702) 293-7426

!! NOW SHIPPING with ALL 'BURNERS !!
TOOLKIT Software for Editing EPROM Information, Plus HEX/MOTOROLA 'S'/Binary File Conversion Utilities

Ultra Violet Products EPROM Erasers
Model DE-4 - Holds 8 EPROMS - Special Prices

Add $3 Shipping & handling. ($2 Bare Boards). C.O.D.'s accepted; Foreign Orders add required postage. Specify Environment and Media Requirements.
H8, H89, Z100 are TM Heath/Zenith; CP/M is TM Digital Research; TRS-80 Model III is TM Tandy Corp; IBM-PC and PC-DOS are TM IBM, MS-DOS is TM Microsoft

Circle #152 on Reader Service Card

ACCESS™ *with Autopilot*™

A New Breed of Communications Program

ACCESS™ *lets you . . .*

- Talk to bulletin boards, information utilities, main frames or other micros
- Send and receive any kind of file—text, binary, .EXE, .COM, etc.
- Transfer files using a variety of protocols, including XMODEM protocol
- Use Autopilot to turn your computer into a communications robot
- Unattended auto-answer mode configures to caller's baud rate, and
- Depending on caller's password, grants unlimited, limited, or no access
- Lets the caller run other programs (data base, editors, etc.)
- Logs password, duration, date and time of calls you receive
- Logs calls you place—recording number dialed, duration, date & time
- Compatible with all modems, even USR S100 and other board modems

How You Can Get ACCESS
Buy ACCESS from your local Heath or ZDS dealer. If there isn't a dealer near you, order it direct from Hilgraeve Inc.

ACCESS, H/Z-89 with CP/M (2.0 or later and 64K req'd) $49.95
ACCESS, H/Z-110 or 120 with ZDOS, CP/M-85, or CP/M-86 . $59.95
ACCESS, H/Z-150 or 160 with MSDOS or CP/M-86 . . . . . $69.95
Add $3.00 shipping; MI residents add 4%

Hilgraeve Inc. • P.O. Box 941 • Monroe, Michigan 48161 • (313) 243-0576

Circle #203 on Reader Service Card

Suppose during the execution of the program (as in procedure PRINT_CHECK, starting on line 173) we wanted to output to the listing device. First we would ASSIGN LP:, the listing device, to BFILE (line 181).

```
ASSIGN(BFILE,'LP:');
```

This links 'LP:' to BFILE. Then it's simply a matter of opening BFILE (line 182) with a REWRITE:

```
REWRITE(BFILE);
```

And then we can output to the printer, as in line 205:

```
WRITELN(BFILE,CHECK.DATE:11,
    CHECK.CKNO:10,CHECK.PAYEE:39);
```

With multiple-drive systems where data files may reside on any of the mounted disks, we can use '*:FNAME' in our program. This will in effect force the runtime system to search all currently mounted disks for the file.

RESET

RESET is the simplest to use of the four operations involved in data-file management. And, with but one exception, it causes few problems for the beginning programmer. That one exception, however, can cause a great deal of head scratching if not properly handled.

If a file is resident on a mounted disk, a RESET(FNAME) will open the file for a READ operation. Also, the end-of-file (EOF) will be set false. The file pointer is set to the beginning of the file.

A READ(FNAME,DATA) will then access the data for the program. Very straightforward and uncomplicated. The problems begin when you attempt to open a file with the RESET procedure when the file is not on any of the mounted disks. The runtime system detects the error, issues a runtime error, and exits the program, returning to the operating system.

Unfortunately, there will be times when files will have to be accessed with a RESET—but during the initial run of the program, they will not yet have been set up on the disk.

However, there is a way to get around this problem. In the runtime system, there is a memory location which contains a byte that controls what happens when access to a nonresident file is requested.

This is at hexadecimal location 22E0 in PRUN. In the documentation, this location is identified as DERRPR. If this byte in the runtime system is set true, an error will be returned. If it is set false, it will return an EOF = TRUE to the program—without the fatal error message and crash. The trick is to write the program such that the program receives and tests for this EOF = TRUE.

First, the default mode of the compiler always returns the error message. So how do we toggle DERRPR to get EOF = TRUE back to the program? This brings us to a fuller discussion of pragmats than we have had so far.

Pragmats

In Chapter 3 of the Polybytes documentation, there is a brief discussion of pragmats. A number of weaknesses of the international standard for Pascal are overcome with the use of these pragmats, or extensions, in this implementation. They are not part of the language, but are included as compiler instructions. Of the five pragmats listed in the documentation, "$A= abs.address" is the one of current interest. $A will establish a variable in the program which can be found at an absolute address in memory (specified in hexadecimal).

Note line 13 in the main body of the checking program of Listing 1. In the record type CHECKS (lines 9-15), the variable .DATE is established; and DATESTRING is assigned to .DATE in lines 26 and 27:

```
(*$A = $20BF*);
DATESTRING : LINE;
```

This is an example of the use of the $A pragmat. These lines establish the variable DATESTRING, a variable of type LINE (defined in line 8). DATESTRING is equal to the values starting at hexadecimal location 20BF in the disk operating system (HDOS). It is a string nine characters long and equivalent to the date currently in HDOS.

The BUFFMODE pragmat

BUFFMODE (lines 24 and 25) is another example of the use of pragmat $A.

Circle #188 on Reader Service Card

Circle #152 on Reader Service Card

Circle #104 on Reader Service Card

Circle #119 on Reader Service Card

BUFFMODE shows my use of LMODE, given in the Polybytes documentation. It is a one-byte location in the runtime system. It changes input from character-by-character operation to line-by-line.

In most Pascal implementations I have seen, input to a program is made on a character-by-character basis. That is, as soon as a character is typed on the console, it is processed by the program.

There is often a considerable disadvantage to this type of input. Let's say that a program input statement is:

READ(NUMBER);

where NUMBER has been previously declared a variable of type INTEGER.

Suppose we intend to input to the program a "1" but slip and hit "Q". The program expected an integer and received a character. All input to a Pascal program must match the type specified by the input statement. In this case, the program doesn't know what it's expected to do with the character, so it bombs.

The best way to avoid this problem is to make all input in the form of characters. (All keystrokes are characters!) But the character must be converted to an integer to be useful to the program. I've seen many "eloquent" procedures for converting characters to integers or reals; all of these procedures take up many program lines.

BUFFMODE is another way to defeat this problem associated with character input. It utilizes the buffered-input facility of the H/Z19 terminal or H/Z89 computer. With the type-ahead buffer, a goof can be corrected, for the program will not see any of the character input until a RETURN is entered.

Lines 24 and 25 of Listing 1 create a variable BUFFMODE of type BOOLEAN (true/false) at runtime location hex 22DF. Any time BUFFMODE is set TRUE (as in line 315 of Listing 1), buffered input into the program is established.

It is with REWRITE, in conjunction with POSITION, that random file access becomes available.

There's another method for accomplishing the same thing as setting BUFFMODE true or false: the use of the pragmat $B in the PRUN command line calling the program.

DEV:PRUN $B DEV:CHECKING

would result in buffering of all input to the program. There are times when character-mode input is desirable, however. So I prefer BUFFMODE to the overall $B usage.

Toggling DERRPR

But let's get back to our original problem—avoiding that fatal error message when we try to read a file that has not yet been set up. To do this, we have to be able to toggle DERRPR true or false. And we can do that by using the $A pragmat and our own Boolean variable DERRPR.

Lines 28 and 29 in Listing 1 illustrate this:

```
(*$A = $22E0*)
DERRPR : BOOLEAN;
```

Here, the initial line establishes that there is a variable in the program which can be found at location hex 22E0. The line immediately following declares DERRPR the name of the variable, and that it is of type BOOLEAN. When we wish to toggle DERRPR false (as in line 86 of the listing), we use

DERRPR := FALSE;

Or we toggle it TRUE (as in lines 93 and 97) by

DERRPR := TRUE;

In both of the examples furnished with the Polybytes documentation, these program statements (lines 24 to 30) were placed with the variables in the main program block.

It was of interest to me to determine if these pragmat declarations could be included in a procedure. A short test program verified that they can. This is particularly useful when developing a "library" of procedures or functions which require pragmat declarations.

Using DERRPR

Let's see how changing the status of DERRPR is used in the checking program. The procedure STARTING_BALANCE, line 84, offers an excellent example.

STARTING_BALANCE opens disk file ACCNTBAL to obtain the balance in the account, the next check number to be printed, and the number of the first check in the active check file.

When the program is initially run, there is no file ACCNTBAL residing on the disk. A person could go crazy trying to make this program work the first time around if the procedure tried to reset a nonexistent file to read in the balance of the account. This is a case where we need some means to disable the return of the system error produced by attempting to open a nonresident file with a RESET.

Line 86 assigns the value FALSE to DERRPR. DERRPR will thereafter return an EOF = TRUE during the initial run of the program. If NOT EOF (BFILE) (line 89) is false (EOF = TRUE), the program skips down to line 95 in the IF... THEN... ELSE... construct. The program then recognizes that this is the initial run. It requests a starting check number for the account from the operator, then returns to the menu.

If EOF= FALSE, as in subsequent runs of the program, then the program executes the IF... portion (lines 90 to 94) of the IF...

THEN... ELSE... statement. It reads the file for the values of BALANCE, CHECKNO, and LATEST in line 91, then returns to the menu. Line 93 sets DERRPR true again.

REWRITE and POSITION

The Polybytes compiler supports sequential files, which have to be read through from beginning to end in order to access particular data. But it also supports files where particular records can be addressed individually. These random files make life a lot easier in a number of applications. That is particularly true in a program like my checking-account program.

REWRITE(FILENAME) is the more versatile of the two file-access opening statements. It is with REWRITE, in conjunction with POSITION, that random file access becomes available to the user.

If you open a file for access with a REWRITE(FILENAME) and the file does not reside on disk, an EOF = TRUE is returned to the program. The next operation on the file can only be a WRITE(LN) to create a file with that name. A runtime error will occur if any other operation is attempted. This can be a sequential write; or one may set up a random file using POSITION.

For instance, a sequential file could be set up as in Example 2. REWRITE opens up FNAME, and SENTENCE is written to it. FNAME can be opened and read with RESET. Example 3 shows the steps in reading a sequential file like this one.

```
SENTENCE := 'Now is the time for all good men.';
ASSIGN(AFILE,FNAME);          (* Link FNAME to AFILE *)
REWRITE(AFILE);               (* Open AFILE for a WRITE *)
WRITELN(AFILE,SENTENCE);      (* Write to the file *)
```

Example 2. Setting up a sequential file with REWRITE.

```
ASSIGN(AFILE,FNAME);          (* Link FNAME to AFILE *)
RESET(AFILE);                 (* Open AFILE for a read *)
READLN(AFILE,SENTENCE);       (* read the record *)
WRITELN(SENTENCE);            (* show it on console *)
```

Example 3. Using RESET to read a sequential file.

```
ASSIGN(AFILE,FNAME);          (* Link FNAME to AFILE *)
REWRITE(AFILE);               (* open the file *)
POSITION(AFILE,0);            (* position pointer to first record *)
READLN(AFILE,SENTENCE);       (* read the record *)
WRITELN(SENTENCE);            (* show it on console *)
```

Example 4. Using POSITION to read a sequential file.

If a file which resides on a disk is opened for access with a REWRITE, then two options are available to the programmer—depending upon how you want to affect the file.

If the next operation is a WRITE(LN), then all previous information contained in the file will be overwritten. It is imperative that the programmer recognize this, or valuable data could be lost.

Lines 331 through 334 illustrate a REWRITE where the previous data in the file was overwritten. Line 331 links 'ACCNTBAL' to BFILE (the file of characters). Line 332 opens BFILE for a write. And line 333 writes back into ACCNTBAL the new values of BALANCE and CHECKNO, and the previous value of

LATEST. The file is closed in line 334.

You can also use POSITION after REWRITE to open for an update by random access. POSITION is the very heart of random file operations, since it is used to set a file pointer to individual records in the file. We can access the file, as in Examples 2 and 3, using the REWRITE procedure and utilize POSITION in a read operation. Reading a sequential file with POSITION is shown in Example 4.

Before a read operation can be performed on a file opened with a REWRITE, the procedure POSITION must be used first to set the file pointer. If this is not done, a runtime error will be returned; the program will abort and return to the operating system.

A better example of POSITION

Actually, Example 4 is a rather poor one for working with random files in Pascal. One of the most attractive features of the language is its rigid data structure and the ease of defining new and more powerful data types—specifically, the type RECORD. Random files are greatly enhanced using RECORDs.

Initially, on the first run of the program in Listing 1, the file of checks, "CHECFILE", does not exist on the data disk. So we run into a problem similar to that discussed with REWRITE. I would receive a file-access error back from the system if I were to attempt to use the POSITION procedure to position the file pointer to the first record in the file (as in line 201, procedure PRINT_ CHECK).

Lines 199 through 204 in the procedure PRINT_CHECK were used in the program to circumvent this problem during the initial run of the program.

If the file does not reside on disk, the system returns an EOF=TRUE back to the program. Then, in line 199, the condition

IF NOT EOF(AFILE)

is not satisfied. So execution skips down to line 204 for the ELSE statement in the IF... THEN... ELSE... construct of this segment of the program. Line 204 simply writes the first record, or check, to the file.

In subsequent runs of the program when line 199 is encountered, the condition IF NOT EOF(AFILE) is satisfied; so lines 200 through 203 are executed. Line 201 positions the file pointer to the next record in the file (CHECK.CKNO-LATEST). And line 202 writes the record to the file, WRITE(AFILE,CHECK). (You should note the ease with which the record "CHECK" was written to the disk file.)

The problems begin when you attempt to open a file with the RESET procedure when the file is not on any of the mounted disks.

The RECORD type CHECKS was declared in the beginning of the main program block, lines 9 through 15. It consists of five separate items of varying types:

1) CKNO, the check number, an INTEGER;
2) PAYEE, an ARRAY OF characters;
3) AMOUNT, a REAL number;
4) DATE, the variable which is assigned the value of DATESTRING which was obtained from the operating system using the $A pragmat; and
5) CKOFF, a Boolean variable which is used in the procedure CHECK_STATEMENT to indicate that the check has been returned.

CHECKS is a five-item record consisting of 48 bytes, all stored on disk with a simple WRITE statement. And it is just as easily retrieved with a READ statement. The only way I can get comparable ease of access to such complex data types from disk files is from a Digital Equipment Corporation MINC computer at my office, using virtual files.

UNPACK

Let's turn our attention now to the procedure PRINTER_READY (line 39)—and to further extensions offered by the Polybytes compiler, notably, "UNPACK", "INPORT", and "EXTERNAL".

I should note here that my system consists of an H89, two double-sided 80-track drives, one 40-track drive, and an Epson MX-80 printer. I'm using a parallel port from FBE Research with printer drivers from both FBE and Lindley Systems.

Therefore, the procedures PRINTER_READY (line 39) and PORTSTATUS (line 64) and the function INPORT (line 45) in Listing 1—which demonstrate UNPACK, INPORT, and EXTERNAL—will not be applicable to a serial port, because the ready and busy bits on a parallel I/O chip are presented differently. However, the general principles of the procedure will be the same. Only the tested bit in the returned status byte from the printer port will change.

In any event, let's start with UNPACK. It differs from what you might expect.

In mainframe implementations of Pascal, the compiler uses horrendous amounts of internal memory to store individual data items. The "PACK" feature of these applications (and of UCSD Pascal) is used by the compiler to store data in a compact way.

After a data item is PACKed, it must be UNPACKed before it can be used by the program. This is the standard use of UNPACK in Pascal compilers.

In the Polybytes implementation, however, "UNPACK" has an entirely different usage. Basically, UNPACK allows you to "unpack" an individual byte into its component bits.

When the printer is ON but OFF LINE, the FBE printer driver will display a reverse-video 25th line message to the effect that the printer is OFF LINE. However, if the printer is OFF during the execution of a program, the message will not be displayed. The program will hang up, waiting for a ready signal from the printer. Often, if you are not paying attention to what's occurring, you may mistake this activity—or lack of it—for a program crash.

What I wanted to do was write a procedure which would test the condition of the printer. It would signal back to the program that the printer was OFF with the same type of reverse-video display on the 25th line. Then it would go into a loop until the printer was turned on.

What this procedure must do is access the printer port with INPORT and EXTERNAL, then UNPACK the returned data byte to determine the state of the printer ready-bit.

The documentation for the UNPACK function is, shall we say, rather skimpy—especially for us "hackers." While initially studying the language and compiler, I skipped over this function since I couldn't at the time recognize its usefulness. You have to think in binary, and even then the documentation can be confusing.

The form of the UNPACK function is: UNPACK(INTEGER,SHIFT,MASK). According to the documentation, this function returns the result of taking the binary value of the first number (INTEGER) and shifting it right by the number of places you specify in SHIFT; then you apply the bit AND operation with the number indicated by MASK. The SHIFT and MASK parameters must be constants.

```
1   PROGRAM BITTEST;
2   (* Program will be called with PRUN BITTEST and PRUN
       $B BITTEST *)
3   CONST
4      SP = ' '; ESC = CHR(27); PAGE = CHR(12);
5   VAR
6      BIT : INTEGER;
7   (*A = $22DF*)
8   LMODE : INTEGER;
9   (*$S *)
10  BEGIN (* Bittest *)
11     WRITELN(ESC,'E',PAGE);           (* Clears the screen *)
12     WRITELN('LMODE byte = ');
13  BIT := UNPACK(LMODE,15,1); WRITE(BIT);
14  BIT := UNPACK(LMODE,14,1); WRITE(BIT);
15  BIT := UNPACK(LMODE,13,1); WRITE(BIT);
16  BIT := UNPACK(LMODE,12,1); WRITE(BIT);
17  WRITE(SP);
18  BIT := UNPACK(LMODE,11,1); WRITE(BIT);
19  BIT := UNPACK(LMODE,10,1); WRITE(BIT);
20  BIT := UNPACK(LMODE,9,1); WRITE(BIT);
21  BIT := UNPACK(LMODE,8,1); WRITELN(BIT);
22  END. (* Bittest *)
```

Listing 2. BITTEST.PAS, a program to test the UNPACK function.

On page 3-24 of the Polybytes documentation, there is an example: UNPACK(256,8,1). This yields the value 1. Here, 256 is the decimal equivalent of the two-byte binary number 0000 0001 0000 0000. When we shift this binary number right 8 places, we allow the right-most bits to fall off the table; and we fill the left-most bits with 0s. So we are looking at the least significant bit of the most significant byte—which when ANDed with 1 does yield 1.

As an exercise to clarify my understanding of this UNPACK function, I wrote a short program to test it. This is BITTEST, shown in Listing 2.

What I wanted to do was select a location in memory whose value I was certain of—and whose value I could change from outside the program without a great deal of difficulty. Then I could manipulate that number with UNPACK to see what would happen.

What immediately came to mind was LMODE (mentioned above in reference to BUFFMODE). LMODE, at hex 22DE in the runtime system, takes on one of two values, depending upon the console-input mode of the program: false (0) for character mode; true (1) for buffered line input.

LMODE also just happens to be a very easy location to change in the runtime system—by using the $B pragmat in the PRUN command line calling the program. The $B pragmat in the PRUN command line changes LMODE from a default value of false (0) in the runtime system to true (1).

What I figured was that if I could display each bit in LMODE, I would get a binary number 0000 0001 when the $B pragmat was used; and binary 0000 0000 when not used.

So BITTEST.PAS was written to test the function UNPACK. Looking at it in Listing 2, you can see that lines 7 through 9

establish a variable LMODE of type INTEGER in PRUN runtime location $22DF. This is the byte which I want to unpack and investigate.

The function UNPACK(INTEGER,SHIFT, MASK) stipulates that the byte to be unpacked must be an integer. Integers are stored in memory as two bytes. When I declare LMODE an integer, I will read into the program two bytes at $22DF and $22E0. BITTEST deals only with the first, or most significant, byte of the two—which is immediately available when LMODE is declared.

Lines 13 through 21 UNPACK bits 8 through 15 of this integer LMODE one at a time. They write the value of the bit, 0 or 1, to the console, WRITE(BIT).

When I ran this program with the PRUN command line, PRUN BITTEST (character-input mode), I received the following output to the console:
LMODE byte = 0000 0000
With a PRUN command line PRUN $B BITTEST, I received:
LMODE byte = 0000 0001
This is what I had anticipated and it satisfied the purpose of the test program.

INPORT and EXTERNAL

We have become familiar with the means of accessing individual bits in a data byte. For our procedure for displaying the printer status, then, the only thing left to determine is how to get the status byte from the parallel printer port.

Tucked away back in Appendix E of the documentation is a brief explanation of a few more extensions offered by the Polybytes compiler—specifically, port I/O routines and disk operating calls.

Port I/O routines include a procedure "OUTPORT" for passing a data byte to a port, and the function "INPORT" used for reading a data byte from a port. What we are interested in for this particular discussion is the function INPORT. (But if you know how to use one of the two, you know how the other works.)

```
FUNCTION INPORT(PORTNUMBER : INTEGER) : INTEGER;
EXTERNAL INPRT;
```

In the PRUN.ACM file supplied on the compiler disk, INPRT is EQUated to PRTIN; at runtime, PRTIN is location 22E4 hex. In a program, INPRT can be declared a constant, such as INPRT = $22E4. Or $22E4 can be substituted for INPRT in the function declaration.

Polybytes Pascal allows the linking of assembly-language programs by following the procedure or function with the declaration:

```
EXTERNAL "absolute address;"
```

A call to this absolute address is executed whenever the procedure or function is referenced in the program. Program control is passed to the absolute address specified in the external declaration.

In Listing 1, FUNCTION INPORT (PORTNUMBER : INTEGER) : INTEGER, lines 45

```
1     PROGRAM TESTPORT; (* Program to test function INPORT *)
2     CONST
3        PORTNUMBER = 208   (* 320 octal, my printer port
                               number *)
4     VAR
5        STATE, READY : INTEGER;
6        RESPONSE     : BOOLEAN;
7     FUNCTION INPORT(PORTNUMBER : INTEGER) : INTEGER;
8     EXTERNAL $22E4;
      (* When the program runs, this function inputs the byte
         representing the current status of the printer port *)
9     BEGIN   (* testport *)
10       REPEAT
      (* Want to go through this test three times, once with the
         printer OFF, once with printer ON, but OFF line, and
         again with the printer ON and ON line *)
11         STATE := INPORT(PORTNUMBER); (* STATE is equal to status *)
                                        (* byte of printer *)
12         READY := UNPACK(STATE,7,1);  (* UNPACK all 8 bits of *)
13         WRITE(READY);                (* STATE and display them *)
14         READY := UNPACK(STATE,6,1);
15         WRITE(READY);
16         READY := UNPACK(STATE,5,1);
17         WRITE(READY);
18         READY := UNPACK(STATE,4,1);
19         WRITE(READY);
20         READY := UNPACK(STATE,3,1);
21         WRITE(READY);
22         READY := UNPACK(STATE,2,1);
23         WRITE(READY);
24         READY := UNPACK(STATE,1,1);
25         WRITELN(READY);
26         READY := UNPACK(STATE,0,1);
27         WRITELN;
28         WRITELN('Again? <Y/N> : ');
29         READLN(RESPONSE);
30      UNTIL RESPONSE = 'N'
31    END. (* testport *)
```

Listing 3. TESTPORT.PAS, **a program to test the** INPORT **function.**

and 46, is an example of the use of EXTERNAL. INPORT is an assembly-language routine, at absolute address hexadecimal 22E4 in the Polybytes runtime system, which will be accessed when this function is called in the program.

Suppose you have an assembly-language driver for one of the high-priced printer-plotters and wanted to access this device from a Pascal program. This driver would be loaded into high RAM, below the operating system, and then referenced by the program with a function or procedure call which will address the starting location of the driver with EXTERNAL.

To make certain that I knew what I was doing with the function INPORT, I wrote another short test program. TESTPORT.PAS reads the condition of the printer port. (See Listing 3.)

In the program TESTPORT, I declared PORTNUMBER (line 3) a constant equal to 208 decimal (320 octal). I used the REPEAT... UNTIL... loop to test the port in the three possible configurations (OFF; ON but OFF LINE; and ON).

Table 1 shows the results when I ran TESTPORT. It's obvious from the results that the two status bits of the printer-status byte are the left-most two, 6 and 7. The documentation which accompanied my parallel port shows bit 6 as the ON LINE bit (high when on line). Yet it also returns a high when the printer is OFF.

Bit 7 shows high when the printer is OFF; and also high when the printer is ON but OFF LINE. This is the bit of the status byte that I decided to key on in procedure PRINTER_STATUS. The printer will not be opened with a REWRITE (PRINTER) until after the printer status has been checked by using procedure PRINTER_READY in Listing 3.

But you may have noticed that STATE, the INTEGER in TESTPORT's UNPACK operation, has only eight bits—instead of the 16 that a Lucidata integer is supposed to have. What happened to the second byte?

When using the INPORT function, the data byte from the port is stored in the least significant byte. At the same time, the runtime system clears the most significant byte of the data integer. In other words, after the INPORT function is executed, the data byte is in the range 0 through 255 decimal.

The PRINTER_READY procedure

With this understanding of UNPACK, INPORT, and EXTERNAL, I wrote the procedure PRINTER_READY in CHECKING.PAS (Listing 1, lines 39 through 82).

In procedure PRINTER_READY, the main body of the procedure is lines 71 through 82. There, line 72 calls procedure PORTSTATUS. PORTSTATUS equates the value of the function INPORT(PORTNUMBER) (line 67) to STATUS. Line 68 assigns the bit value of the UNPACK function UNPACK(STATE,7,1) to READY, then returns to the main procedure at line 73.

Printer configuration	Output
ON	0111 1111
OFF BUT ON LINE	1011 1111
OFF	1111 1111

Table 1. The results of the TESTPORT.PAS program, identifying the two status bits of the printer-status byte.

In line 73, IF the bit value of READY is high (1) THEN lines 75 through 81 are executed. Or ELSE the procedure falls through to the end. (Which indicates that the printer is on and on line.)

Line 75 calls procedure TWENTY_FIFTH_LINE (lines 48 to 62), passing the value of READY, which is high (1). TWENTY_FIFTH_LINE contains a case statement whose execution depends upon the value of READY. Line 59 executes if READY is 1. This simply enables the 25th line of the console; then returns to the main portion of the procedure at line 76. Line 76 prints the message "Printer is OFF...or printer is OFF LINE!!!" on the 25th line of the console.

Lines 77 through 79 set up a loop which calls PORTSTATUS until the printer is turned on. Then the loop exits to line 80. Line 80 calls TWENTY_FIFTH_LINE, passing READY with a value of 0. The 0

disables the 25th line on the console and restores the cursor to its original position on the console screen.

With the reverse-video display on the 25th line, the user is never uncertain about the status of the program—so far as the printer is concerned.

In conclusion

Pascal has been criticized for its verbosity. This is, unfortunately, quite true. Granted, also, the compiler discussed in this article is slow. The lack of speed was also pointed out by Dr. Kenner in *Sextant* #6.

But most of us can live with these faults. The one thing I would recommend to the author of the runtime system, Mr. Reeve, is an in-line editor. It could be such that the compiler would be diverted to the editor on an error. After the error was corrected, compilation would continue from the point of the error.

Also, I would like to take this opportunity to point out a mistaken impression that some may have about the Polybytes compiler and Pascal in general from comments offered by Dr. Kenner in *Sextant* #6. There, he asserted that Pascal does not support dynamic arrays such as are available in BASIC. Strictly speaking, that may be true. But we can get around this limitation—by using Pascal pointer types.

However, pointer types are not an easy concept to grasp unless one is familiar with assembly-language programming. One of the truly fine presentations on the subject is given by Dr. Elliot B. Koffman in *Pascal, a Problem Solving Approach,* Addison-Wesley Publishing Company.

It is my own opinion, however, that pointers have few really good applications on small 8-bit systems. (That statement should create a fuss!)

Polybytes' Lucidata Pascal compiler and runtime system by Larry Reeve is undoubtedly a very fine software-development system available to the Heath/Zenith computer community.

If you are running BASIC and would like to get involved in a truly fine learning experience, I would certainly endorse Pascal, and the Polybytes implementation.

Ordering Information

Polybytes Lucidata Pascal Compiler, $55.
P-Code Translator, $25.
Specify hard-sectored or soft-sectored 5" CP/M or HDOS; or 8" CP/M.
Polybytes
3427 Bever Avenue, S.E.
Cedar Rapids, IA 52403
319/366–3077

Getting the Most Out of an Okidata Printer Under Z-DOS

You can configure your Okidata u92 to your specifications from a menu-driven program.

M. H. Endres

As we all know, merely connecting Product A to Product B does not guarantee a working system. Pieces of it may work. Parts of it may function as expected, while other advertised features may be impossible to implement.

Luckily, such is not the case with the u92 printer from Okidata. After the exercise of building my H100 computer from a kit, the interfacing of the Okidata u92 was simplicity itself.

I chose to interface the printer through my '100's parallel port. I did so in order to save the '100's two RS-232C serial ports for other planned uses. My cable was a stock RS-232C "Centronics compatible" cable as sold by Heath (Heath # HCA-20). I used the Zenith Disk Operating System (Z-DOS) CONFIGURation program.

No initial problems were encountered with the computer/printer communications. I could print using either PRINT or COPY under Z-DOS. Printing a file from my word processor was no problem, either.

The problem came when I tried to send a file to the printer under Z-BASIC with the LLIST command. The first attempt at LLISTing a program from Z-BASIC resulted in over-typing the perforations along the left side of my green-bar printer paper.

Reading the Okidata manual, it became clear that upon initialization, HOME or Position 1 will locate the print head at the left-most end of travel. And the tractor feed cannot be moved far enough to the left to avoid printing over the perforations when one wishes to use inexpensive 8½"-wide green-bar paper for listings. So the left margin must be changed before beginning any Z-BASIC LLIST or LPRINT operations.

In response to this irritating state of affairs, I offer the short program in Listing 1.

It is true that a suitable printer configuration may be accomplished by typing the appropriate LPRINT statements in Z-BASIC command mode before listing any documents. But anyone with day-to-day requirements for the computer and printer will know that it is much faster and more convenient to have a short configuration program to do the work for you.

The program presented here allows you to select typeface, printing quality, and page width. It is designed to reside on the same disk as Z-BASIC, and is the first program run when Z-BASIC is utilized. The program is annotated, structured, and error-trapped to prevent erroneous data from being entered.

It is much faster and more convenient to have a short configuration program to do the work for you.

The program

The first segment of the program selects the typeface desired (i.e., pica, elite, or condensed). The second segment selects either data-processing mode (*fast-print mode* at 160 characters per second) or correspondence mode (*slow-print mode* at 40 characters per second).

Data-processing mode is used for program listings, data output, and other in-house uses. Correspondence mode provides excellent letter-quality printing by printing each line twice: a small offset between printings effectively eliminates the "dots" from the dot-matrix print-head!

The third program segment sets up the left margin of the Okidata printer to any desired point from position 001 (HOME) to position 999—with each position being approximately 1/120 of an inch farther right. I've found that for 8½" green-bar paper, centered on the carriage, a left margin data entry of 050 will provide a good starting spot within the border of the greenbar.

This program segment ensures that *any* width paper (within the limits of the machine) may be used with the Okidata tractor feed to accept LLIST and LPRINT outputs from Z-BASIC.

And PeachText

Many H/Z100 owners utilize the PeachText 5000 word-processing system. That's not surprising, since Heath/Zenith sells the package to use specifically with the H/Z100. This article was written using the PeachText package—which I have found to be exceptional in both documentation and ease of use.

Unfortunately, the Okidata series of printers is *not* one of the printers supported by PeachText's printer-configuration software. We Okidata users must be satisfied with using the printer in the general mode called "DRAFT PRINTER."

This, by itself, means that specialty printer options (which the Okidata printer has in abundance) are not *directly* available using PeachText commands. Please note that *"directly,"* for there is indeed a way "OUT."

The PeachText command word "OUT" may be used to modify the Okidata printer's default configuration. The OUT is used to embed control characters in text files. It takes the form OUTn, where "n" is the ASCII decimal equivalent of the desired control character. So all we need to do is put in our files the characters which the Okidata will recognize as controlling the various printer options. When the file is printed, the options will be activated.

When we are in PRINT MODE, for example, OUT12 will output an ASCII

```
10 REM OKIDATA MICROLINE 92 CONFIGURATION PROGRAM
20 REM USED TO SET UP PRINTER PRIOR TO USE WITH
30 REM GREENBAR PAPER OR WITH WORDPROCESSOR TO SET
40 REM MODES AND TYPE FACE.
50 REM
60 REM BY M.H.ENDRES 12-13-84
70 REM
80 CLS
90 PRINT "PLEASE SELECT TYPE FACE DESIRED"
100 PRINT "1. 10 CHARACTERS PER INCH (PICA)"
110 PRINT "2. 12 CHARACTERS PER INCH (ELITE)"
120 PRINT "3. 17.1 CHARACTERS PER INCH (CONDENSED)"
130 PRINT
131 PRINT "DEFAULT (RETURN) = 10 CHARACTERS PER INCH"
132 PRINT
140 INPUT "SELECT 1,2 OR 3";A$
141 IF A$ = "" THEN A$ = "1"
150 IF VAL(A$) < 1 OR VAL(A$) > 3 THEN PRINT
    "MUST BE 1,2,3 OR RETURN" :FOR X = 1 TO 1500:NEXT:GOTO 80
160 GOSUB 1000
170 CLS
180 PRINT "SELECT DESIRED OPERATIONAL MODE"
190 PRINT
200 PRINT "1. DATA PROCESSING MODE (160 CHARACTERS PER SECOND)"
210 PRINT "2. CORRESPONDENCE MODE  (40  CHARACTERS PER SECOND)"
220 PRINT
230 PRINT "DEFAULT (RETURN) = DATA PROCESSING MODE"
240 PRINT
250 INPUT "SELECT 1 OR 2";A$
260 IF A$ = "" THEN A$ = "1"
270 IF VAL(A$) < 1 OR VAL(A$) > 2 THEN PRINT
    "MUST BE 1,2 OR RETURN":FOR X = 1 TO 1500:NEXT:GOTO 170
280 GOSUB 2000
300 CLS
310 PRINT "SET LEFT MARGIN"
320 PRINT
330 PRINT "DEFAULT (RETURN) = 'HOME' OR POSITION 1"
340 PRINT
350 INPUT "DESIRED LEFT MARGIN -- 001 TO 999";A$
360 IF A$ = "" THEN A$ = "001":GOTO 380
365 IF LEN(A$) < 3 THEN PRINT "YOU MUST ENTER ALL THREE NUMBERS":
    FOR X=1 TO 1500:NEXT:GOTO 300
370 IF VAL(A$) < 1 OR VAL(A$) > 999 THEN PRINT
    "MUST BE BETWEEN 001 AND 999":FOR X=1 TO 1500:NEXT:GOTO 300
380 GOSUB 3000
999 END
1000 REM CHARACTER PER INCH SELECTION
1010 A = VAL(A$)
1020 IF A = 1 THEN B = 30 ELSE IF A = 2 THEN B = 28 ELSE B=29
1030 LPRINT CHR$(B)
1999 RETURN
2000 REM OPERATIONAL MODE SELECTION
2010 A = VAL(A$)
2020 IF A = 1 THEN B = 48 ELSE B = 49
2030 LPRINT CHR$(27);CHR$(B)
2040 RETURN
3000 REM LEFT MARGIN SET
3010 LPRINT CHR$(27);CHR$(37);CHR$(67);A$
3020 RETURN
```

Listing 1. Program to set up an Okidata Microline 92 printer. This program moves the left margin, and sets the print quality and number of characters per inch, as specified by the user. Lines 150, 270, 365, and 370 have been broken to fit the space here. They should be typed in as one line each.

"FF" or FORM FEED to the printer. By utilizing the Control Code Listing in the Okidata manual and the OUTn PeachText command statement, the printer may be controlled *character by character* as to typeface, spacing, double-width, emphasis, enhancement, and graphics-mode selection. The appearance of your printed copy is up to you.

I must emphasize, however, that the OUTn command works only during the actual print process. Should you type an OUTn at the backslash prompt when viewing PeachText's PRINT MENU, *nothing* will happen with the printer. When RETURN is again struck, that OUTn command will be the first item output to the printer. It will cause the desired FORM FEED—but not necessarily when you want it!

The real power of the OUTn command is that it is most convenient to use as an embedded command beginning and ending with a backslash (\OUTn\) within the text itself. In this fashion, you need not fret over when the command

```
\SETUP
\LM6,RM6,BM6,JUST
\FORMFEED OFF
\TEXT
\OUT27,49\This selects correspondence quality mode (40 CPS)!
\OUT27,48\This selects data processing rate ( 160 CPS)!
\OUT28\This is 12 character per inch Elite type.
\OUT29\This is 17.1 character per inch compressed mode.
\OUT30\This is 10 character per inch Pica type (normally used for most purposes).

This line shows how \OUT31\DOUBLE WIDTH \OUT30\characters can be embedded in a line

How about \OUT27,84,27,72\EMPHASIZED \OUT27,73\words?

Would you like superscript \OUT27,74\tm   \OUT27,75\printouts?

How about sub\OUT27,76\scripted \OUT27,77\words?
```

Figure 1. A demonstration file using the PeachText OUT command to embed printer-control characters in text.

```
This selects correspondence quality mode (40 CPS)!
This selects data processing rate ( 160 CPS)!
This is 12 character per inch Elite type.
This is 17.1 character per inch compressed mode.
This is 10 character per inch Pica type (normally used for
most purposes).

This line shows how DOUBLE WIDTH characters can be
embedded in a line

How about EMPHASIZED words?

Would you like superscript tm   printouts?

How about subscripted words?
```

Figure 2. The codes embedded in the text in Figure 1 result in this output on an Okidata printer.

will be executed. It will execute when your document is printed out—at whatever point in the text you placed it.

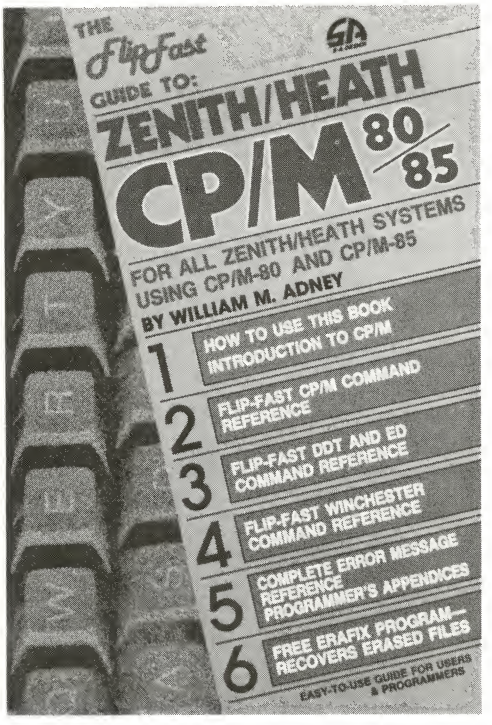
Figure 1 shows how a simple demonstration file would look when you saw it on the screen under PeachText. Figure 2 shows that file printed out on my Okidata.

Putting it together

It is easy, then, to start your documents, letters, or whatnot with an appropriate command list (setting margins, justification, centering, initiating form feeds to ensure starting off with a clean sheet, etc.).

Further, you will find that within a few days' use, a "standard" will emerge for different types of documents. This, in turn, will lead you to the creation of document "shells" and a beginning library of "standards" with which to further improve speed and productivity when generating, editing, and printing documents. But that's up to you!

Circle #102 on Reader Service Card



Finally!

A Convenient Guide to CP/M-80 and CP/M-85 Written Specifically for Users of the Heath/Zenith '8, '89, '90, '110, and '120.

The FlipFast Guide to Zenith/Heath CP/M 80/85

Published January 1984
by S-A Design Books
190 pages

Spiral bound so pages turn easily and lie flat—lets you flip around quickly to find the information you need, then stays open flat to the page you want.

Convenient 5½″ × 8½″ size leaves you more room next to your computer than standard binders.

You'll find the following information for each CP/M command or subcommand:

- The required **syntax** with available options
- Its typical **applications**
- Specific **examples** with explanations of command functions
- A listing of command **requirements** for correct/best operation
- **Application notes** about specific usage
- **Error message** listing of command-specific error messages

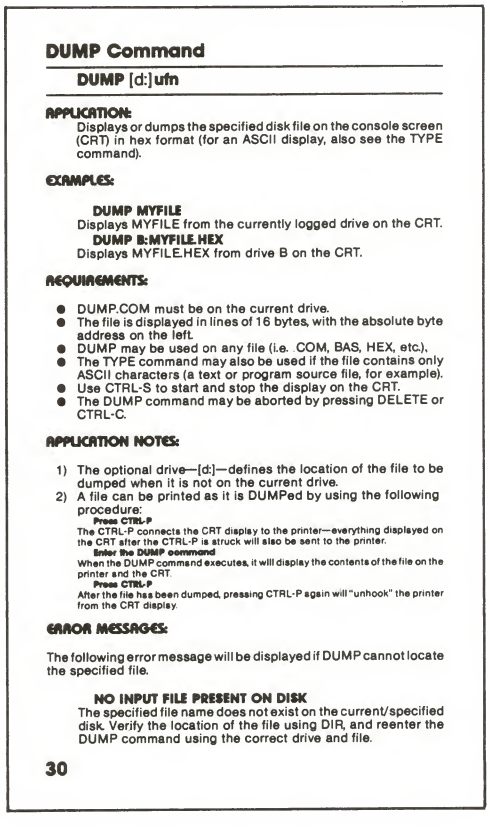
And you can easily refer to listings of:

- CP/M 2.2 system calls 0–40
- BIOS jump table
- 8080/8085 op codes in both alphabetic and numeric order—with decimal, hex, and ASCII equivalents
- Escape sequences and key functions of the Heath/Zenith '19, '89, '90, '110, and '120

Plus:

Additional technical information on the Disk Parameter Block, Disk Parameter Header, File Control Blocks, CP/M memory addresses for various memory sizes, and the contents of the first page in memory.

Features a 6-page index of error messages.



DUMP Command

DUMP [d:]ufn

APPLICATION:
Displays or dumps the specified disk file on the console screen (CRT) in hex format (for an ASCII display, also see the TYPE command).

EXAMPLES:

DUMP MYFILE
Displays MYFILE from the currently logged drive on the CRT.
DUMP B:MYFILE.HEX
Displays MYFILE.HEX from drive B on the CRT.

REQUIREMENTS:

- DUMP.COM must be on the current drive.
- The file is displayed in lines of 16 bytes, with the absolute byte address on the left.
- DUMP may be used on any file (i.e. .COM, .BAS, .HEX, etc.).
- The TYPE command may also be used if the file contains only ASCII characters (a text or program source file, for example).
- Use CTRL-S to start and stop the display on the CRT.
- The DUMP command may be aborted by pressing DELETE or CTRL-C.

APPLICATION NOTES:

1) The optional drive—[d:]—defines the location of the file to be dumped when it is not on the current drive.
2) A file can be printed as it is DUMPed by using the following procedure:
Press CTRL-P
The CTRL-P connects the CRT display to the printer—everything displayed on the CRT after the CTRL-P is struck will also be sent to the printer.
Enter the DUMP command
When the DUMP command executes, it will display the contents of the file on the printer and the CRT.
Press CTRL-P
After the file has been dumped, pressing CTRL-P again will "unhook" the printer from the CRT display.

ERROR MESSAGES:

The following error message will be displayed if DUMP cannot locate the specified file.

NO INPUT FILE PRESENT ON DISK
The specified file name does not exist on the current/specified disk. Verify the location of the file using DIR, and reenter the DUMP command using the correct drive and file.

30

Circle #246 on Reader Service Card

CONTENTS

CONTENTS—(Continued)

Circle #120 on Reader Service Card

Circle #231 on Reader Service Card

Circle #211 on Reader Service Card

MBASIC Marquee

Two lines of programming can turn your screen into an electronic billboard.

Raymond Dotson

It's always a challenge to try to get the maximum of activity out of the minimum of programming. Some time back, I saw a useful little program that seemed to do just that. So, of course, I decided to see if I could improve upon it.

The program was by Roy Reichert and called "Electronic Billboard." (It appeared on the back cover of the Heath Users' Group magazine, *REMark*, issue #17, May 1981.) In the Electronic Billboard, a series of character strings were entered by the user. Then the program displayed this data on the screen in running marquee fashion, right to left. The program used only 20 lines to do the job.

What I present here is a two-line program which is, in effect, the same program as Mr. Reichert's.

The new program is called MARQUEE and its activity and capabilities belie its short length. Otherwise static and lifeless computer displays may be livened by running this extremely short program with appropriate string entries.

As you use it, it's as simple as typing in whatever you want to appear on the screen. You may enter your message as one continuous string of as many as 256 characters.

You may also enter it as two or more strings which will be joined together by the program. As you are entering them, be sure to remember to enter one or more spaces between these strings. Series of periods (. . . .) are also very effective at the end of a message to be displayed.

All two lines

Looking at Listing 1, you can see that I use lines longer than the normal terminal width. When you enter the program, you can break lines with LINE FEED. True, the program could have been written with one statement per line. But why not take advantage of MBASIC's ability to handle multiple statements per line? Compactness of programming and increased execution speed make this one of the nicer features of MBASIC over older BASICs. This is a feature that is now copied by most other BASICs.

Line 100 clears the screen and asks for numbered string inputs (called "lines" by the input prompt). (It will take no more than 256 of these "lines.") These strings will be joined together to form one string labelled B$. RETURN on a blank line will terminate this input phase and cause the program to progress to line 200.

(If you have your terminal set up for a 256-character width, you can input the entire message on one "line." It may look funny as it appears on the screen. But the carriage returns thrown in on the screen won't appear in your message.)

Line 200 turns off the cursor, clears the screen, and uses direct cursor addressing to print the left-most 20 characters of B$ in the center of the screen. (You can change that, of course, if you want a longer or shorter display.)

The left-most character of B$ is removed and added to the right end of B$. There is a short delay to permit proper viewing time; then the program is sent back to perform line 200 all over again.

All the while, the keyboard is being monitored for any key strike. The space bar will restore the cursor and end the program; all other keys are ignored.

Other BASICs

As it's presented here, MARQUEE is written in Microsoft BASIC version 5.1 to run under the CP/M operating system, version 2.2.03. With minor changes, however, this program will work perfectly well under the Heath Disk Operating System (HDOS) in either MBASIC or Benton Harbor BASIC.

For either, omit the portion of line 200 starting with "A$=INKEY$" and ending with "ELSE". This will remove the keyboard monitoring and automatic cursor restoration when the space bar is hit. You will be required to terminate operation with a Control C and then manually restore the cursor (off line, ESC y 5).

The "90" in the FOR... NEXT... loop of line 200 will need to be reduced to compensate for the slower execution time of B.H. BASIC.

The maximum length of B$ is the same (256) in either of these BASICs. So therefore you may input many short strings or a single long one—just so the maximum length is 256 or less.

However, when using HDOS MBASIC, add a new line as follows: "10 CLEAR 256". This will permit input string lengths of more than 33 characters.

Try MARQUEE. I think you'll find a use for it—perhaps a program that could be livened up a bit—perhaps you could just run it to have your computer display a greeting at a party.

Maximum activity with minimum programming....

```
10 C$=CHR$(27):PRINT C$"E"C$"Y,1LINE"I+1;:LINE INPUT "  ";A$:I=I+1:
       IF A$<>"" THEN B$=B$+A$:GOTO 10
20 PRINT C$"x5"C$"E"C$"Y,="LEFT$(B$,20):
       A$=INKEY$:IF A$=" " THEN PRINT C$+"y5":END ELSE
       B$=MID$(B$,2)+LEFT$(B$,1):FOR U=1 TO 90:NEXT:GOTO 20
```

Listing 1. MARQUEE lets you write a message and display it repeatedly moving across the screen. The message can be up to 256 characters long. Written for MBASIC under CP/M, the program can be modified for B. H. BASIC and MBASIC under HDOS.

Scuttlebutt at HUGCON 3

HUG pulled off another CON, despite the best efforts of some of the staff of Pheasant Run Resort.

Well, another HUG conference has come and gone. The Third International Heath/Zenith Users' Group Conference was held July 27–29 at Pheasant Run Resort in St. Charles, Illinois. Several *Sextant* staffers were there. We talked to lots of subscribers and advertisers and authors, and saw a number of new products for Heath/Zenith computers and users.

Pheasant Run Resort is quite some distance from O'Hare Airport. When the hotel's shuttle bus failed to pick up some *Sextant* staffers at the appointed hour, we were forced to resort to a limousine. (Life is tough in the big time). It turned out, though, that the limo cost less for three riders than the bus would have.

The jokes began as soon as the first attendees made their reservations. Looking at the prices, one HUGgie said "they should have held it at *Peasant* Run." Unsatisfied with some of the facilities, one of our staffers replied, "it *was.*"

Even with the high prices, though, Pheasant Run filled up fast—and apparently overbooked. We heard rumors of people with confirmed reservations being sent to another hotel seven miles away, and being turned away there, too.

We didn't see many HUGgies around the pool and hot tub. But some *Sextant* staffers availed themselves of every opportunity to rest themselves at poolside. Apparently everyone else was at the Heathkit Electronic Center counter shopping for bargains. (We didn't have a chance to check out the golf course—maybe that's where the rest of the estimated 1,300 attendees were.)

Of course, there's always room for complaints regarding the facilities at conferences like this. And Pheasant Run was nowhere near an exception. Many who *were* able to get a room had complaints about comfort. The exhibit area was half hot, half cold, and too small. There were frequent complaints that some of the staff at the resort were unhelpful. (Yet others went out of their way to be nice.) And the orange juice at breakfast both days tasted rancid.

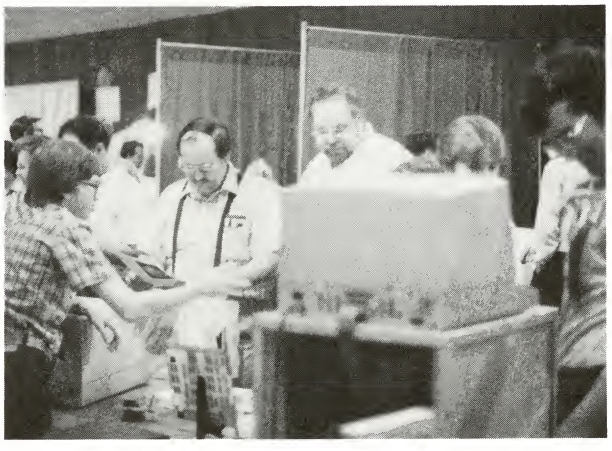


Photo by Charles Floto

The view from *Sextant*'s booth. It was hard to see beyond SigmaSoft & Systems' booth, which attracted many interested H89 users.

The poor service from some of the Pheasant Run staff was not the only subject of complaints. Many attendees found that the Heath/Zenith representatives were sending HUGgies an unwelcome message: that the IBM-compatible market is Zenith's future. In his keynote speech Saturday morning, Phil Cole, director of product planning at the Heath Company, said that Zenith is moving into the future with their IBM-compatible products. And the grand prizes awarded Saturday evening were two H150s and one Z150. (About 75 prizes from independent vendors were also awarded.)

We thought Phil Cole's keynote speech was summed up best by Joseph Gonzalez in the Puerto Rico HUG newsletter. "Apparently [he] thought that we HUGgers do not read our Heath catalogues. He had a slide show to tell us that Heath now handles D-G, Magnolia, etc. Frankly, the powdered eggs I got for breakfast excited me more than the speech. At least I had not seen powdered eggs since World War II, while the Heath catalogue I had seen the week before."

Bob Ellerton opened the ceremonies at breakfast on Saturday. Bowing to tradition, he brought up a subject dear to a lot of hearts: "Yes, Virginia, there will be HDOS 3.0."

He presented some awards: Outstanding User—Bill Adney, author of *The Flipfast Guide to Zenith/Heath CP/M-80/85;* Outstanding Club—Capital Heath Users' Group, for the second year in a row; Outstanding Vendor—Walt Bilofsky of The Software Toolworks, the first software publisher for Heath/Zenith computers; Lifetime Membership—H.W. Bauman, a frequent author for *REMark;* and From User to User—Bill Parrott, who originally conceived of the National HUG Conference.

We attended several of the conference sessions. The Ruby Room was packed on Friday night for Walt Bilofsky of The Software Toolworks, orating on the C language. Susan Hayes, also of the Toolworks, outdrew Walt, though, at her session on Computer Chef. So now we know what Heath users are *really* interested in.

On Saturday morning the vendor area didn't open until 11:30. So after the 8:30 breakfast there was lots of competition for seats at the sessions. One of us wanted to hear Barry Watzman talk about multitasking operating systems, but couldn't get in the room.

Bill Parrott and Dale Wilson answered questions about the long-awaited HDOS 3.0 operating system Saturday afternoon. HDOS 3 will be available for the '110 and '120 as well as for the H8 and H/Z89. It will support hard disks and 64K of RAM.

There will be some changes in the HDOS 3 directory. One addition is a 'Delete Protect' flag which prevents a file from being erased, but doesn't block file updates. Another is a record of the number of accesses to a file, up to 255.

Despite these and other alterations to the code, Parrott promised an extremely high degree of compatibility

with software written to run under HDOS 2. Parrott said that HDOS 3 may be out by the end of 1984. (Is it here?)

Bill Adney spoke on the differences between CP/M-80/85 and CP/M-86 Saturday afternoon.

Sunday morning, Mike Cogswell held forth on Z100 hardware. We've seen him do this before—last year, at CHUGCON. He knows his stuff, and answered lots of questions.

The vendors

There were some new faces among the vendors, and some familiar faces in new contexts. Barry Watzman, former computer product line manager for Heath Company, exhibited for the first time, selling Concurrent CP/M for the Z150 and CP/M-86.

Dysan Corp., the disk company, exhibited their Interrogator Drive Diagnostic Program. This was their first time exhibiting at a Heath/Zenith conference.

Software Wizardry had two booths—the second was for their dealership, First Capitol Computer. They introduced their superfloppy drive in a Z100—2.7 megabytes on a 5¼" floppy. They also demonstrated their Palette graphics program connected to a digitizing tablet.

M.P.I. was there to show their printers—with the Zenith Data Systems name on the front! One of the printers was a Sprinter—a 16-pound portable.

Apparently the Z160 "portables" were *just* hitting the market the weekend of HUGCON. Studio Computers was showing some, but Software Wizardry had not yet received any.

Husker Systems of Nebraska has a new corporate name: NewAmerica Technologies Corp. (Does this mean we won't see the Nebraska Husker quarterback on their stationery any more? Or will he just be playing for a different team?)

C.D.R. Systems showed their new Z100 speed module. The ZS100 is externally switchable between 5 and 7.5 megahertz. (See "Supplier Notes," *Sextant* #13.)

Innovative Data Systems was selling software at discount all weekend long. By Sunday, the prices were down to $25 and under for packages worth hundreds of dollars. Needless to say, they didn't have to ship any of their inventory back home after the conference.

There was a crowd around SigmaSoft and Systems' booth all weekend. The center of attention was their Interactive Graphics Controller/Pseudo Disk for the H8/19 and '89. This board adds high-resolution graphics, a semiconductor memory system, two trackball/joystick interfaces, and two Centronics parallel ports.

The prize for "Heath/Zenith user-compatible product" has to go to Fusaro Associates of Farmingdale, New York. They were displaying several "Backsaver" ergonomic chairs. We don't know how many chairs they sold, but there was no shortage of weary HUGgies willing to try them out.

Once again, Heath/Zenith suppliers travelled from all corners of the country to ply their wares. There were at least six companies from California. (They're always overrepresented, though.) We counted three exhibitors from Texas. Vendors also came from Arizona, Maryland, Missouri, New York, North Carolina, Utah, Rhode Island, Louisiana, and New Jersey.

The users proved themselves to be the most dedicated, though. HUGgies travelled from as far away as Israel, Australia, South Africa, and West Germany.

All in all, this year's HUG Conference was a first class production on the part of the HUG staff. And it showed the dedication of many HUGgies—that they came and endured despite inadequate accommodations and lack of helpfulness from the hotel staff.

We had fun, though, We can only hope the next HUG Conference has fewer barriers to surmount.

AFFORDABLE COMPUTER COMPONENTS, EQUIPMENT, SUPPLIES AND SERVICES

A REFURBISHED DAISY WHEEL PRINTER
FOR PERSONAL COMPUTER USERS AND SMALL BUSINESSES.

Three-In-One Offer! Just $598 (Includes On-Site Warranty) †

- A 30 cps **letter-quality** printer
- A timesharing keyboard terminal (when modem equipped)
- A Selectric*-style keyboard typewriter

AJ daisy wheel printer terminals are renowned for exceptional performance, high reliability, and applications versatility. Now you can have all this for only $598** in our special limited offer.

- 30 cps *letter-quality* printing
- Changeable type faces
- Full ASCII keyboard with numeric pad
- High resolution X-Y plotting
- Complete electronic forms control
- 128-character buffer
- Asynchronous RS-232 interface
- Printwheel, ribbon cartridge, and cable included
- 30-day parts/labor warranty

And you can choose from a list of options including forms tractor, pin-feed platen, paper trays, side shelves, extra printwheels, APL keyboard and 2K buffer.

For information telephone

(215) 279-3320

†Within 50 miles of service center.

**Suggested selling price, excludes options and is subject to change without notice. Model shown includes certain options. Offer available only in the contiguous U.S.

*Selectric is a trademark of IBM.

ACCESS

206 DeKALB STREET
NORRISTOWN, PA 19401

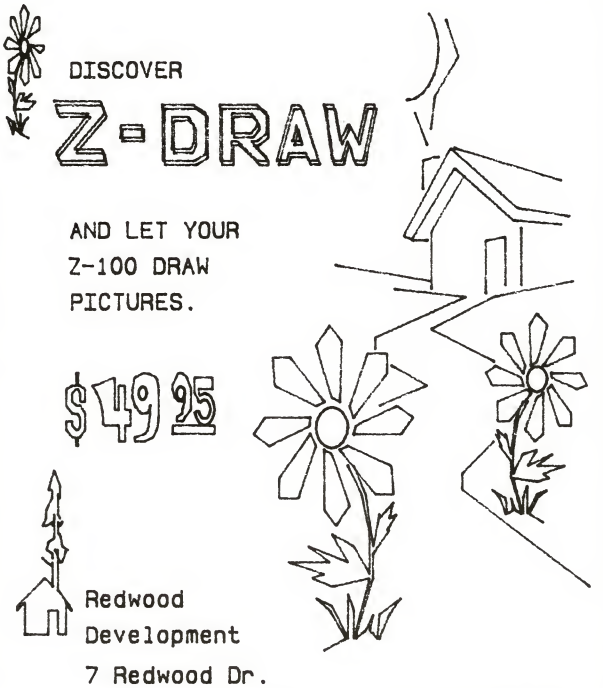
Z100*
CP/M PLUS
is now available.

For further information, please contact:

Barry Watzman
560 Sunset Rd.
Benton Harbor, MI 49022
616/925-3136

*Z100/110/120: not for Z150/160 series.
CP/M Plus is a trademark of Digital Research.

Circle #206 on Reader Service Card



Circle #196 on Reader Service Card

USE ALL YOUR SPECIAL FUNCTION KEYS

With WORDSTAR™

WSKEY™: Now you can take the mystery out of WordStar with SKILL DATA's WordStar enhancement, which implements all twenty-one of the H/Z89-19 function/pad keys or all twenty-three Z-100 labeled key commands.
Function key commands are labeled by a twenty-fifth line banner, which can be toggled on and off by you during your session.

With dBASE II™, ZIP™, SuperCalc™

DBKEY™, ZPKEY™, and SCKEY™: Just type your favorite SKILL DATA key command. All function key commands are labeled by a twenty-fifth line banner, which can be toggled on and off by you during your session. Pad keys also function and send multiple key inputs with a single stroke. All previous command key sequences are still available for the old and painful ways.

Just $29.95 for each program (plus shipping charges shown below).
H/Z89-19 diskettes are 5.25" 10HS, require CP/M 2.2.03.
Z-100 diskettes are 5.25" SS; specify CP/M-85 or Z-DOS.

Available in many Heathkit Stores and Quikdata Inc., Sheboygan, WI...or complete and mail this order form with your check or money order today!

Mail to SKILL DATA, P.O. Box 1943, Olympia, WA 98507.

Yes, I want to use all my special function keys! Send me:

☐ WSKEY—keys to WordStar: ☐ for H/Z89-19 ☐ for Z-100 $29.95
☐ SCKEY—Keys to SuperCalc (for CP/M only) $29.95
☐ DBKEY and ZPKEY—keys to dBASE II: ☐ for H/Z89 ☐ for Z-100 $29.95
For Z-100 diskettes, specify operating system: ☐ CP/M-85 ☐ Z-DOS

Name _____
Street _____
City _____ State _____
Zip _____ Phone _____
Visa/MC Card # _____ Expires _____

All orders shipped by first class mail. Include $3.00 per order for shipping and handling, $5.00 for non-U.S. orders. Payment by check, money order, or Visa/MasterCard. Allow 2-4 weeks for delivery. 206/352-0669 (evenings). (s)

SKILL DATA, P.O. Box 1943, Olympia, WA 98507

DATAFLEX™

THE COMPLETE DBMS
INTERACTIVE SCREEN OPERATIONS.
"IMAGE" FORMATTING
POWERFUL COMMAND LANGUAGE

"NOW"
For
Z-90
Z-100
Z-150

Applications created with DataFlex have:

FLEXIBILITY -
EXPANDABILITY
DATAFLEX MENU
SPEED and EFFICIENCY
REPORT GENERATION
MULTI-USER CAPABILITIES
"FLEX-KEY"™ COMMAND ENTRY

EEI data systems

P.O. BOX 55-8487
MIAMI, FLORIDA 33255 (305) 559-4401
A DIVISION OF ERNEST'S ELECTRONICS, INC.

DataFlex and Flex-Key are trademarks of Data Access Corp.

Circle #168 on Reader Service Card

Focus on your system, with *How to Use: Zenith/Heath Computers*

Get the guide to your microcomputer system—a 144-page book by Hal Glatzer, specifically for the Zenith/Heath user.

Serving as both an introduction to computing for the novice, and a reference tool for the expert, *Zenith/Heath Computers* features:

- ideas to help you develop your system
- a survey of software compatible with your hardware
- reviews of peripheral products such as:
  disk drives—expansion interfaces—print spoolers—modems—letter-quality and dot-matrix printers
- a glossary, an index, and a list of suppliers supporting your system
- photographs and illustrations to clarify technical information

This is not a complicated users manual!
Published December 1982 by S-A Design Books.

ORDER YOUR COPY TODAY!
Simply fill out this coupon now, and send it to:
Sextant Publishing Co., Dept. S14,
716 E Street S.E., Washington DC 20003 • 202/544-0900

☐ YES! Send me ___ copies of *How to Use: Zenith/Heath Computers*
at $19.95 each = _____
☐ Payment enclosed (checks must be in U.S. dollars payable on a U.S. bank)
☐ Charge my: ☐ Visa ☐ MasterCard
Card # _____ Expires _____
Send via: ☐ UPS ☐ Parcel Post Special Handling

Name _____
Address _____
_____
_____

Please allow 4-6 weeks for delivery.

Circle #191 on Reader Service Card

---

why not . . .
GIVE YOUR COMPUTER A JOB AROUND THE HOUSE?

ByteSize Concepts, Inc. INTRODUCES—

THE HOME SERIES

HOME BUDGET MANAGER
*Calculator built-in
*Statement reconciliation
*Check printing with auto feature
*Search features

HOME MECHANIC
*Complete cost breakdown per car
*On-line trouble-shooting guide
*Over eighty 'pages' of HOW-TO help
*Easy menu control of all functions

HOME DATA MANAGER
*Household inventory
*Credit card management
*Address, phone, recipe lists
*User defined data bases
*Easy to use features

plus $2.75 S+H **$49.95**
Ohio residents add 6.5% sales tax each

Complete Package: All Three Only $120.00

ORDER TODAY! ORDER TODAY! ORDER TODAY!

ByteSize Concepts, Inc.
18101 Euclid Avenue
Cleveland, OH 44112

Circle #136 on Reader Service Card

New Generation Brings the Charm of Unix to CP/M

MicroShell and MicroTools give you Unix-like utilities at CP/M prices.

Walter G. Jung

If you're a microcomputer user and have used Digital Research's CP/M disk operating system for any length of time, you've noticed its strengths and its weaknesses. The first strengths we usually see, of course, are its low cost, standardization, and a powerful software base which includes many business-oriented applications.

The weaknesses we frequently think of are in the "friendliness" department. Its ambiguous error messages may be the first thing that the new user notices; but there are other ways in which CP/M can be hard to work with. Some of CP/M's limitations, however, are simply a product of its age. CP/M was first offered in 1976. For microcomputers, that's ancient history.

A number of products are offered to keep CP/M current with the latest developments. We'll take a look at two of these products: MicroShell, authored by Rick Rump; and Donald Graft's MicroTools. They're both offered by New Generation Systems of Reston, Virginia.

Some background: redirection and SUBMIT files...

But before we go into MicroTools and MicroShell, it might be well to take a closer look at CP/M and at some of the newer ideas that MicroTools and MicroShell intend to bring to CP/M.

MicroShell and MicroTools both bring to CP/M some of the virtues of one of the most powerful operating systems around: Bell Labs' Unix. Unix is not yet feasible in the world of ordinary 8-bit microcomputers. But it involves two ideas that are. These are "redirection" and "software tools."

"Redirection" refers to the input to or output from a program—not merely data, but also the commands that control the program.

Ordinarily, the user provides input from the keyboard. Where a program's output is sent is similarly directed from the keyboard during operation. Redirection allows you to control input and output by means of another program, a disk file, or the operating system's command line. Input redirection occurs when a program receives its commands from a disk file, for instance; output redirection allows the program to pass its output to a file. When the output from one program is input directly to another program, the redirection process is called a "pipe."

Users of the Heath Disk Operating System (HDOS) already have a limited form of redirection. A disk directory may be sent to the printer or to a file, for instance, with CAT LP:=SY1: or CAT SY1:FILENAME.EXT=SY1:.

All programs which use standard CP/M system calls should work.

CP/M has the edge over HDOS in another element of a more complete redirection capability, however. CP/M's SUBMIT capability already provides the user with the ability to control multiple program execution from a disk file. A SUBMIT file is intended to contain a series of commands that would otherwise be typed in at the keyboard while operating at the operating-system level.

A SUBMIT file might be as simple as the name of your editor followed on the next line by the name of the text formatter you use for printing. Running the file would call up your editor. Then while you're working with the editor, CP/M keeps track of where it is in the SUBMIT file. When you exit the editor, CP/M calls up the text formatter.

The SUBMIT capability can be further expanded by the use of argument substitution. You can assign a number of "dummy" arguments to a command line in the SUBMIT file: $1, $2, and so forth. When the SUBMIT function is run, you are able to assign values to those arguments. CP/M's XSUB program, when called on the first line of a standard SUBMIT file, allows you to substitute keyboard input for the dummy arguments. (With a command line such as EDITOR $1, the argument might be a device and file specification.) This is, of course, input redirection *to* a program.

Under the Zenith Disk Operating System (Z-DOS), the SUBMIT capability is referred to as batch processing. (You might want to take a look at *Sextant* #10, May-June 1984, for "Standard Operating Procedure: Take Advantage of Z-DOS Batch Files," by Peter Norton.)

The SUBMIT capability is an important feature of the Unix environment and of MicroShell. But it gets discussed in a somewhat different terminology. What would be called a batch or SUBMIT file under Z-DOS or CP/M is called a "*shell file*" under Unix and MicroShell. And what would be called batch or SUBMIT processing, under Z-DOS or CP/M, is called "*shell programming*" instead. Those are the terms that will be used in this review.

The difference is not just terminological, however. SUBMIT and XSUB, as useful as they are, are still limited basically to what you might type in at the CP/M command line. With complete redirection, you can also provide input to a *second program's command line* from a SUBMIT-type program. This ability to redirect input to a program extends the power of the SUBMIT capability considerably. And that is not all.

...and "software tools"...

If redirection is one popular attraction of Unix, the idea of "software tools" is the other. Here, too, HDOS users are familiar with a limited form of the idea.

Under CP/M, communication with peripherals such as printers, disk drives, etc., is handled through CP/M's Basic Input/Output System (BIOS). Any time a new peripheral is added, code to drive it must be incorporated into CP/M's BIOS. This can make life easier for the user: if a peripheral is supported, talking to it can be readily arranged by means of CP/M's CONFIGUR utility (provided by Heath/Zenith).

But the more peripherals you have, the harder it becomes for the programmer to avoid possible conflicts while "fitting" everything into the BIOS.

Circle #172 on Reader Service Card

Circle #210 on Reader Service Card

Circle #146 on Reader Service Card

Circle #224 on Reader Service Card

```
a) MicroShell v2.0 files:

Name     Ext Bytes   Name     Ext Bytes   Name     Ext Bytes   Name     Ext Bytes
AUTOCPM  COM    1K ! FULLPRMPSUB    1K ! SH       COM   10K ! TIMBUS  COM    1K
D        COM    3K ! MENU     SUB    1K ! SH       OVR   11K
ERASE    SUB    1K ! NORMPRMPSUB    1K ! SHSAVE   COM    1K
10 File(s), occupying 31K of 238K total capacity
54 directory entries and 207K bytes remain on A:


b) MicroTools v1.2 files:

Name     Ext Bytes   Name     Ext Bytes   Name     Ext Bytes   Name     Ext Bytes
CAT      COM    8K ! DEFORM  COM    6K ! MERGE    COM    8K ! SORT     COM    8K
CHECKSUM        1K ! DIFF    COM    9K ! NEXT     COM    5K ! SPL      COM    6K
COL      COM   10K ! ECHO    COM    4K ! P        COM    7K ! STR      COM    6K
COM      COM    6K ! FIND    COM   10K ! PASTE    COM    8K ! TEE      COM    4K
CRCK     COM    2K ! GET     COM    4K ! PR       COM   13K ! UNIQ     COM    6K
CRYPT    COM    6K ! GREP    COM   15K ! REC      COM    6K ! WC       COM   10K
CUT      COM    7K ! LN      COM    5K ! SLEEP    COM    4K
27 File(s), occupying 184K of 238K total capacity
37 directory entries and 54K bytes remain on A:
```

Listing 1. The distribution file directories for MicroShell and MicroTools.

HDOS avoids this problem by relying on device drivers. These utility programs are written to handle just one peripheral. Then a driver can be "tacked on" to HDOS as needed, simply by being called from the command line.

Unix, too, relies on the idea of accessing a variety of small programs as needed. But the power dwarfs that of a simple device driver. The intent in Unix is to build a "repertoire" of small utilities. Useful on their own, they can be mixed together in ways that multiply their power. This is also like the idea found in a number of programming languages—where you build libraries of subroutines that can be used in a number of programs.

Mixing the various simple tools into more complex ones, though, can be an art form unto itself. Programming under Unix is not being pushed as an activity for the novice.

(If you wish to look further into Unix, *Byte* magazine had a special "C"-language/Unix issue in August 1983. "The Unix Tutorial," by David Fiedler, was continued in the September and October issues; and the October issue also highlighted Unix.)

...and more

There is a third feature associated with newer developments (Unix among them) that *is* useful to the novice. Elaborate directory programs are now available to cope with the increasing number of files possible on disks with ever increasing space available, such as on hard disks (Winchester drives).

And sophisticated directories prove particularly useful for a CP/M feature that is frequently used by those dealing with novice users—USER areas.

CP/M makes life easier by allowing you to set up a USER area on disk, and by restricting directory access to just that area when the user is logged into it. It eliminates complexity. But it can also be frustrating; if you wished to allow a number of USER areas to access certain programs, you would have to duplicate those programs in each area, with a resulting waste of disk space. A sophisticated directory program can provide the solution.

If, however, Unix has garnered a lot of attention, most microcomputer owners may find it a rather academic alternative. It only runs on computers with fancy central-processing units (CPUs), huge amounts of disk storage, and megabytes of random-access memory (RAM).

And Unix is expensive. CP/M on the H/Z110 and '120, for instance, can cost $100 or less. Even a stripped-down version of Unix for a microcomputer can cost hundreds or thousands of dollars.

Speaking practically, a better step to improvement might be to find software that can help CP/M along. What we need is a "shell" program: one that would stand between us and CP/M, one that would improve the user interface and allow such features as I/O redirection.

MicroShell and MicroTools

MicroShell is such a program, and it can be teamed up with MicroTools. Together or separately, they allow many Unix-like enhancements to run under ordinary CP/M.

MicroShell alone will give you shell-file capability, I/O redirection, and advanced batch processing. MicroTools is a set of Unix-like utility programs. Alone, it will give you redirection. Put the two together, and you get the best of both worlds—not Unix, but more than plain-vanilla CP/M allows.

The distribution file directories for both of these packages are shown in Listing 1, and are discussed below. (The neat directory displays of Listing 1 were captured under MicroShell via D, a public-domain directory program supplied with MicroShell. In this case, the program's output was redirected to a file.)

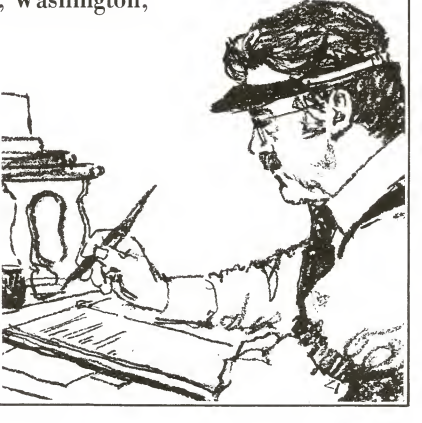
Circle #220 on Reader Service Card

Circle #209 on Reader Service Card

Circle #154 on Reader Service Card

Authors:

Have you considered writing an article for *Sextant*? We're looking for human-interest stories, in-depth technical features, program descriptions, product reviews, and tutorials which relate specifically to Heath/Zenith systems. We pay more than many other microcomputing magazines. Send your article, idea, rough draft, or outline to: John Walker, Technical Editor, Sextant, 716 E Street S.E., Washington, DC 20003.



MicroShell is a completely new "front end" or "shell" for CP/M 2.2 on the H8 and H/Z89 and CP/M-85 2.2 on the '110 and '120. In effect, MicroShell replaces the console command processor (CCP), adding many Unix-style operating enhancements while doing so.

The CCP is that portion of CP/M most apparent to you as a user. It provides the interface between your keyboard commands and the system itself. In its most raw and un-adorned form (what you in fact get with Heath/Zenith's CP/M-80, or CP/M-85), CP/M's CCP is *not* a very friendly system. In fact, it can be a source of irritation to some users completely new to its use. The CCP is responsible, for instance, for such nuisances as having the TYPE command scroll a text file off the screen before you can read it.

Importantly, MicroShell increases CP/M's ease of use, and user friendliness. It requires virtually "zero installation" before it's ready to run. And it is usable not only on the Heath/Zenith CP/M implementations (on the H8, the H/Z89 and '90, and the H/Z100), but on just about all CP/M-80 systems. (Program particulars are listed in Table 1.)

MicroShell improves CP/M in the friendliness department by providing a number of features. First, it retains all of the standard and familiar CP/M-80 CCP commands we all know and love. They operate essentially just as before (DIRectory, ERAse, TYPE to screen, SAVE memory sectors to disk, REName, and USER n).

There are, however, some differences and some enhancements. For instance, you can still TYPE a file to the screen as in CP/M-80. But the additional TYP command pages the screen in 23-line chunks, for readability. And the USER command allows access to up to 31 (instead of 15) user areas. Further, not only do you have access to more user areas, the current one is displayed as part of the prompt (unlike CP/M).

A minus is the lack of query (All?) when you erase all the files on a disk with an ERA *.*. That's not a major penalty, in my opinion. (For those who might disagree, I suggest you consider getting some specially designed erase utility to make the entire job easier. The public-domain program ERAQ is one that can be used.)

Finally, there is a major plus for users of Heath/Zenith's CP/M. You retain all of the very significant operating features which Heath/Zenith included in their current realization of CP/M's BIOS. They include type-ahead buffering, unconditional warm boot (break key), and the flexibility to use peripherals of your choice (H17, H37, hard disk, printer selection, etc).

(In fact, it is worth noting the type-ahead-buffered keyboard and "break" features. Combining them with MicroShell provides a powerful asset to any application program; and it makes word processing particular fun. But I'm ahead of myself!)

a) MicroShell:

Program Reviewed:
MicroShell version 2.0. A shell program with Unix characteristics for CP/M-80 version 2.2.

Company:
New Generation Systems, Inc.
1800 Michael Faraday Drive, Suite 206
Reston, VA 22090
(703) 471–5598
(800) 368–3359

Dealers:
Various CP/M software distributors, or direct.

Price:
$150

Operating Systems:
CP/M-80 version 2.2 compatible systems

Memory Requirements:
64K recommended, will operate in less (see text)

File Package:
SH.COM program and related support utilities, various related demo files.

Disk Formats:
Standard 8" SS/SD format, as well as various 5" formats, including Heath hard- and soft-sectored.

Documentation:
Approximately 55 pages including index, plus various appendices, in a three-ring binder.

b) MicroTools:

Program(s) Reviewed:
MicroTools version 1.2, a utility software tool program set, for operation with Unix characteristics under CP/M version 2.2. For use either in conjunction with and as a complement to MicroShell, or as stand-alone utilities.

Company:
MicroTool Software
P.O. Box 12
Naperville, IL 60566

Dealers:
New Generation Systems; see above.

Price:
$150

Operating Systems:
CP/M-80 version 2.2 compatible systems.

File Package:
CAT, COL, COM, CRYPT, CUT, DEFORM, DIFF, ECHO, FIND, GET, GREP, LN, MERGE, NEXT, P, PASTE, PR, REC, SLEEP, SORT, SPL, STR, TEE, UNIQ, WC.

Disk Formats:
As above.

Documentation:
Approximately 100 pages including a tutorial discussion, as well as detailed descriptions of each individual utility. Packaged in a three-ring binder.

Table 1. Program particulars for MicroShell and MicroTools.

A shell between you and CP/M

MicroShell is, as the name implies, a "shell" program. As such, it attempts to operate as a shell emulating Unix (within the limits of CP/M, of course). On disk, it is 10 kilobytes in size; in RAM, it takes up 8 kilobytes in addition to standard CP/M. It loads itself into memory and replaces 2K of the operating sys-

Circle #195 on Reader Service Card

Circle #235 on Reader Service Card

Circle #111 on Reader Service Card

Circle #110 on Reader Service Card

tem's true CCP. There, MicroShell processes commands and passes them to CP/M.

For those new to this shell concept, the main point to remember about MicroShell is that it takes control of input and output (I/O) for *all* programs which run under it. (See the discussion in the comments accompanying this article, "Some MicroShell Practicalities".)

Controlling I/O between programs and the operating system has some very important implications. One is that *any* CP/M program can be used with redirected input and output features—*whether or not it was originally designed to do so.*

A second consequence is that since MicroShell functionally replaces the user-to-CP/M interface, it allows an entirely new operating environment to be created. As it turns out, the new environment is by far a more friendly one. (No accident!) In fact, it acts almost as if it were a different operating system merely using CP/M's basic system and command language. All this, despite old plain-vanilla CP/M-80 version 2.2 at the root!

Preliminaries

The "main business" part of the MicroShell package is the shell program itself, which is SH.COM. When it is first invoked from CP/M, it signs on, and prints its default prompt. This is shown as the first example in Table 2.

Example	Prompt string	Prompt	Comment
1	-p"%nD(%u) "	A(0) ■	Default
2	-p"%n%n>"	> ■	Simple
3	-p"^[x4%n%n^[p%D%u>^[q ^G"	A0> ■	Uses H19

Table 2. Three prompts under MicroShell. Example 1 is the default prompt. You can change it with the -p command, as shown in Examples 2 and 3.

MicroShell's prompt is distinguished from the standard CP/M prompt (A>). The default MicroShell prompt is "A(0)". The letter A represents the drive you are logged in on; and the number 0 corresponds to the USER area you are logged in on. (Score one in the friendliness department. Ordinary CP/M doesn't prompt with the USER number when you are running from one of the USER areas. It leaves you in the dark, with a single prompt, whether you are in USER 0 or USER 10!)

If you don't like this, you can easily customize it, with the "-P" command, as shown in Table 2. (The "-", by the way, usually indicates either a reversal of a function or, as in this case, a change made to a default.)

The second example in Table 2 yields a plain vanilla ">" (a la HDOS). The third produces "A0>", in reverse video (with a trailing bell, to grab your attention).

You can obviously use your imagination to advantage to customize a prompt. I personally prefer the reverse video and block cursor of example 3; but suit yourself. Just make it different from CP/M, as a reminder that you are in the shell.

As for the "installation" of MicroShell, there isn't much necessary at all. The work is done by SH.OVR, an overlay file which works in concert with SH.COM; it is needed for such functions as customizing MicroShell to your system. All you have to do is customize the shell for the string used on your terminal to clear the screen. In MicroShell, this is referred to as "%clrstr".

MicroShell's default clear string is control-Z. And under MicroShell, the caret (^) key is always used for CTRL functions. To customize MicroShell for the H/Z100 and for the H/Z19 terminal (and for the '89), tell it to use the string "ESCape E". The ESC function is a control-left-bracket so you use the command

Index to Advertisers

Reader Service #		Page #
	ACCESS	
*	Letter-quality printer.	73
	Advanced Software Technologies	
210	Z100 file transfer program.	76
210	Utility software.	98
	American Computer	
188	Hardware for Z100 and Z150.	55
	Analytical Products	
*	Utilities and enhancements for H19 and H89.	25
	Apogee Software	
231	Game.	68
	Array Software	
121	Eight-bit software.	86
	AwareCo	
130	BASIC debugging software.	60
	***Buss* Newsletter**	
107	Heath/Zenith news 20x/year.	35
	ByteSize Concepts	
136	Home management software.	74
	C.D.R. Systems	
108	CP/M BIOS modifier.	53
110	H8 disk controller.	80
111	Z100 speed module.	80
109	H/Z89 disk controller.	95
	CEC	
150	Disk drives.	58
	Cleveland Codonics	
112	Graphics boards.	33
	Colorworks	
149	H/Z100 software.	70
	ComputerTutors	
156	Programming utility for H/Z89/90.	42
155	Game for H/Z89/90.	101
	CompuView Products	
223	Tektronix 4010 graphic emulator for Z-DOS.	77
	D.E.L. Professional Systems	
158	8087 speed and power booster for the Z100.	24
178	IBM-PC emulator board for the Z100.	back cover
	DelSoft	
102	WordStar function-key utility.	65
	Diamond Data	
160	Checking record-keeper for the Z100 and Z150.	70
	Dysan Corporation	
143	Z100 drive diagnostic program.	13
	EEI Data Systems	
168	Utility software.	74
	Eigenware Technologies	
125	Compilers, editors, tools, and other software.	64
	FBE Research Co.	
116	H/Z89 hardware enhancements.	88
	First Capitol Computer	
221	RAM chips for the H/Z100 and H/Z150.	96
	Fisher, Troff & Fisher	
169	Computer law services.	92
	Floppy Disk Services	
117	Half-height and hard drives.	2
	Generic Software	
119	Financial software for CP/M-80 and MS-DOS.	55
	Headware	
192	Tax preparation software for H8, H/Z89, and H/Z100.	92
192	Public domain software directory.	96
	Paul F. Herman	
172	Graphics software for the H/Z100.	76
	H & H Computer Enterprises	
126	Disk storage multiplier units for H/Z89/90.	54
123	Serial/parallel interface; printers; software; hardware.	88
	Hilgraeve	
203	Communications software.	53
	H.M. Technologies	
174	RAM boards for the H/Z150.	8
	Hoyle & Hoyle Software	
205	Tax preparation software.	44
	HSC	
122	16–bit attached resource processor.	72
	Hudson & Associates	
138	8087 system support board.	40
	Bob Huff	
176	Math text software for the H/Z100 and H/Z150.	42
	Interaction Dynamics Systems	
135	Key remapping, general ledger, graphing, flight simulation software.	92
	Interdiscipline	
179	Action games for H/Z100.	99
	Intuitive Logic	
177	Educational software.	69
	Jay Gold Software	
120	Home finance system.	68
	Kandueazy Computer Software Services	
141	Dust covers, tape-to-disk conversion utility.	52
	KEA Systems	
175	Z100 smart terminal emulator.	61
175	Z150/160 smart terminal emulator.	83
	Kres Engineering	
195	H/Z89–90 expansion board.	80
	K's Computerware	
182	Tax preparation software.	51
	LabSoft	
124	Clinical laboratory data-processing software.	56
	Lindley Systems	
132	Ten Mbyte Winchester for the H/Z100 and H/Z150; software.	22
131	Printers and supplies.	62
	Magnolia Microsystems	
134	Network controller, Winchester subsystems, DD floppy subsystems, CP/M Plus.	46
	Mako Data Products	
154	H/Z89–90 all-in-one accessory bus expander.	78
	MDG & Associates	
183	Disk formatter for the H/Z150.	15
	Microservices	
209	H/Z100 and 150-PC graphics software.	78
	Mockingbird Data Systems	
242	Assembler and linkage editor for CP/M.	43
	Mullen Computer Products	
211	Controller and extender boards for the Z100.	68
	New Generation Systems	
214	Disk formatter for S–100 systems.	57
	Newline Software	
145	Software.	9
145	Text processor.	48
234	Tax preparation software.	95
	Northwest Computer Algorithms	
104	LISP for H89.	55
	Larry W. Oberholtzer	
187	Screen dump for H/Z100.	86
190	Evolution simulation program for Z-DOS.	99
	Omni Data Systems	
201	Zenith computers, software, and accessories.	98
	Payload Computer Services	
147	Hardware, software, supplies.	30
	The PC & J Graphics Co.	
146	Printers, supplies, programming aids.	76
	Piiceon	
244	RAM board for the H/Z100.	81
	Redwood Development	
196	Graphics software for the H/Z100.	73
	Reichert Digital Systems	
151	Personal accounting system.	92
	Ross Custom Electronics	
152	Programmer for EPROM, EEPROM, 8748, 8751.	53
152	Programmer for EPROM, EEPROM.	55
	San Diego Scientific	
238	Accounting program for Z-DOS, MS-DOS, PC-DOS, CP/M 80/86.	12
159	H/Z100 speedup hardware.	32
181	H/Z100 and H/Z150 hardware.	62
	SciMed Analysis	
245	XY-plotting software.	50
	Secured Computer Systems	
157	16K memory expansion card; serial and parallel I/O, real time clock for H/Z89.	70
	Sextant	
*	Subscriptions.	19
103	Back issues.	91
	Sextant Publishing Co.	
246	Book on Heath/Zenith CP/M 80/85.	66
191	Book on Heath/Zenith computers.	74
	SigmaSoft & Systems	
185	High-resolution graphics controller and pseudo-disk system.	29
	Sigma Technology Co.	
247	Z-DOS communications software.	86
	Sirius Software	
212	H/Z100 software.	101
	Skill Data	
*	Function key utility.	74
	S & K Technology	
230	Z-DOS text processor.	96
	Skycastle Computer Products	
218	Graphics/text formatter, calligraphy programs.	61
	Soel Publishing	
*	Address book software for H/Z100.	26
	The Soft Firm	
219	File transfer utility for CP/M 80/85 and Z-DOS.	98
	SoftShop	
162	Utilities, languages, educational software.	90
	Software Services of Arizona	
198	CP/M home finance program.	98
	Software Support	
164	Disk drives and printers.	36
	The Software Toolworks	
165	C compiler for CP/M.	59
248	CP/M spreadsheet.	88
	Software Wizardry	
221	Programmable sound, speech, time card for the H/Z100.	16
	Studio Computers	
249	Z150 software, hardware.	14
	Sunflower Software	
171	Z150s, games, utilities, time management software.	95
	Systems Innovations, Inc.	
220	Cartridge backup for Z100.	78
	TEACO	
224	Portable floppy disk tester.	76
	Technical Micro Systems, Inc.	
173	Z80/8086 upgrade for H/Z89.	6
	Barry Watzman	
206	CP/M–86 for the Z100.	47
206	Z100 CP/M Plus.	73
	Westcomp	
235	Z-DOS software.	80
	Wizard Software House	
228	Graphics package for H/Z100.	89
	Workman & Associates	
229	Soft-sectored software for Z90, H/Z100, and H/Z150.	34
	Zeducomp	
236	Structured programming tool.	100
	Zenith Data Systems	
133	Printers.	10

*Contact advertiser directly.

"%clrstr "^[E" ". This will patch the shell in memory. You then should run SHSAVE, to write out the new SH.COM to the disk.

This is the minimum necessary to get MicroShell going; and these (or other) customized parameters will be saved. They will later be recalled whenever SH.COM is invoked.

If you did not have a Heath/Zenith system, you might want to use MicroShell's AUTOCPM utility. This is used to autoboot your CP/M system, and run SH on a cold boot. It is not necessary under Heath CP/M, however, since you can simply run CONFIGUR, and make SH your cold-boot command line. (Be sure to run CONFIGUR from CP/M, not under MicroShell, as I explain in "Some MicroShell Practicalities.") Of course, if you do not wish to have SH.COM auto-execute when you boot up, you can disregard the above.

Search paths

A major enhancement over CP/M-80 is that MicroShell gives you a *search path* for file names input on the command line. It has a search path that will be active for file types with extensions such as .COM, .OV?, and .SUB. What does this mean? In practical terms, it means a lot, so read on!

Under MicroShell, you can be logged into any drive and/or USER area; and a command specifying a .COM file will trigger an automatic search for the program, if it is *not* in the current directory. MicroShell will first search the current user area on the current drive for a .COM file, then USER 0 on the current drive, then drive A:/USER 0.

It does this by default for all the file types noted above, not just .COM types. Not finding a .COM file, it will go back and look for a .SUB file with the same name, and so on. You can, however, change or add to these file types if you wish.

You can also specify other drives for the search path. In addition to (or in place of) the A: drive default, you can specify up to a maximum of seven drives. Don't go overboard, however, since the search takes time. Three seems to be a happy practical limit.

What's the advantage? Simply, you need only *one* set of .COM files for the entire system, including a hard disk, and over all user areas. With this feature, you can put all of your utilities on USER 0 of your hard disk (or your largest floppy) and let MicroShell find them for you.

MicroShell and MicroTools allow many Unix-like enhancements to run under ordinary CP/M.

Note that MicroShell also searches for .OV? file types. This proves useful with a program like WordStar that expects to find its overlay files in the drive you are logged in on. Now you can use it from any user area no matter where it is stored.

In the event that MicroShell can't find a "COMMAND.COM" to execute on the first search through all drives and user areas, it then shifts its gears. As noted, it searches again, this time for a "COMMAND.SUB". This brings up perhaps one of the major features of MicroShell, the built in "SUBMIT-like" batch-processing capability.

As indicated above, a MicroShell "shell file" is a file which contains commands to be handled by MicroShell's batch processor. A shell file always has the filetype (extension) .SUB. It can contain any of the standard commands intrinsic to the CCP. As well, it can contain commands unique to MicroShell and executable program commands (.COM file names).

These commands may all be used singly, or mixed together as a batch process. They are called from the MicroShell prompt. So long as there is no "FNAME.COM", a .SUB file is called just as if it were a .COM file—simply by typing the "FNAME" of "FNAME.SUB". Whereupon MicroShell will load the .SUB file into its buffer, and sequentially execute the series of commands. (This is basically similar to CP/M's SUBMIT; but as we'll see, there is much more.)

As noted, MicroShell's search hierarchy defaults to a "search .COM then .SUB" sequence. So if you have more than one or two drives, it will take time to find a .SUB file for execution. You can override the multiple-disk .COM search, just by prefixing a command for the .SUB file with the drive, such as "B:FNAME". (Later, I will discuss some examples of .SUB files and shell programming.)

Redirection

As noted above, MicroShell also supports many features of the Bell Labs Unix operating system, while all the while operating under CP/M. For example, both input and output redirection are supported, as well as "pipes."

With input redirection, a program may receive commands from a file in

Please refer to the key to the right when completing the article evaluation form on the attached Reader Service Card above. By letting us know how interesting and/or helpful you found each feature in this issue of *Sextant*, you will help us choose the best articles for you in the future.

1. David R. Felstul, **Microsoft Word and Mouse: Do They Click?**, p. 7.
2. Wayne Rash, Jr., **TMSI's H-1000: Grafting an 8086 Onto Your '89**, p. 17.
3. Beaufort Lancaster, **WatchWord: Custom Built for You and Your Z100**, p. 23.
4. Mark Goodkin, **Hugh Kenner's Book for the Z100 Beginner**, p. 31.
5. Roy Coleman, **Put HDOS Commands on Disk**, p. 37.
6. David R. Felstul, **Heathkit Hackers at Home**, p. 41.
7. James Bingham, **Let Your '89 W.I.S.E. Up**, p. 45.
8. Donald E. Risher, **File Management Under Lucidata Pascal**, p. 49.
9. M. H. Endres, **Getting the Most Out of an Okidata Printer Under Z-DOS**, p. 63.
10. Raymond Dotson, **MBASIC Marquee**, p. 69.
11. **Scuttlebutt at HUGCON 3**, p. 71.
12. Walter G. Jung, **New Generation Brings the Charm of Unix to CP/M**, p. 75.
13. Arthur L. Thomas, **Two Debugging Tricks for MBASIC**, p. 97.

```
+-F           File Search:                          On
+-G           Gobble Line Feeds:                    On
+-M           Mode - End of File(+=CP/M,-=UNIX):CP/M
+-T           Transparency Character Flag:          Off
+-U           Upper Case Command Translation:       On
+-V           Verbose: Echo shell files:            On
+-P "string"  Prompt String: %n%D(%u)
-D 1 NNN      Input Character Delay:    0
-D 2 NNN      Input Line Delay:         0
-D 3 NNN      Output Prompt Delay:      0
-D 4 NNNNN    Input Ignore Factor:      0
%path  "str"  Disk Drive Search Path:A
%ftype "str"  File Types-Auto Search:SUB OV? COM
%shext "str"  File Type -Shell Files:SUB
%sdrive char  Scratch Drive-Shell Files: A
%clrstr "str" Screen Clear String:      ^Z
%break on/off Break during Shell Files: On
%bchar  char  Break Character:          Esc
%exchar char  Program Exit Character:   None
%tchar  char  Revert to Keyboard Char:  None
%schar  char  Revert to Shell File Char: None
```

Listing 2. The -s command will give you a status report of the current shell conditions. This example is for the default SH.COM, before any customization.

```
; (This file adapted from MicroShell MENU.SUB for demo purposes)
; Sample Menu Program -- Be sure %clrstr is set for your system.
-v
%start
%A$ = ""
%erase
%print "        ^[pDemo Menu^[q"
%print
%print " 1 - Directory of disk"
%print
%print " 2 - Run CP/M program"
%print
%print " 3 - Exit to CP/M"
%print
%print -n "     Enter selection:"
%getchr %A$ <$T
%erase
%if %A$ = 1 then goto dodir
%if %A$ = 2 then goto runprog
%if %A$ = 3 then %exit
%goto start
%dodir
        d $usv
        %print
        %print
        %print Directory of all USER areas (+ system files, attributes):
        %print
        %goto wait
%runprog
        %print -n "Enter program name [argument]:"
        %getstr %0 <$T
        %print
        $0 $1 <$T
        %goto wait
%wait
        %print
        %print -n "Press any key to continue:"
        %getchr %A$ <$T
        %print
        %goto start
```

Listing 3. DEMOMENU.SUB prints three menu items on the screen, and awaits keyboard input.

lieu of keyboard input. And output can be passed from a program to a file (output redirection). Or output can be passed on to another program, that is, "piped." Also, multiple commands can be stacked on a command line, separated by ";" delimiters.

Input redirection can take a file of commands and input it to an executable program. This could take a form such as:

COMMAND <FILE.CMD

For example, "COMMAND" might be "WS" to call up WordStar. And "FILE.CMD" would be the series of WordStar commands needed to print a file; and would have been edited appropriately for the program accepting input. (A later example shows this type of processing.)

Output redirection is invoked with a form such as:

COMMAND >+FILE.OUT

In this case, COMMAND.COM is the program whose output is being captured; and FILE.OUT will contain the output data. The "+" is an option used for output to the screen as well as to a file, if desired. Output redirection to a file is a powerful asset, both for creating documentation and for programming.

A couple of print-related redirection options are particularly useful. MicroShell, of course, includes CP/M's standard CTRL-P to echo the screen to the LST: device. Additionally, it can redirect output from a program to the printer with a command such as COMMAND>+$P, where the "$P" indicates printer redirection. The command can be given in either uppercase or lowercase. (And the "+" is the screen-output option again.)

A special option is the use of redirection to a file for later printing. For example,

COMMAND >* PRINT.FIL

The * redirection symbol will take the output from a program (a text formatter, for example) and send it to the file for later printing under CP/M.

MicroShell can redirect output not only from .COM files, but also from the MicroShell intrinsics, such as DIR. For example,

DIR >DIR.LOG

will run the directory and create a file (DIR.LOG) for it.

Another file can also be appended to this one by using a double output redirection symbol. For example, DIR B:>>DIR.LOG would add a second drive to the directory example above.

MicroShell has a pipe feature, which allows sequential program outputs to be chained together, to execute a series of commands. The form of the command is COM1|COM2|COM3, etc. The output of COM1 is piped to COM2, the output of COM2 to COM3, and so on. The "|" character tells MicroShell to internally perform the piping process. It does so with the appropriate "<", ">" equivalents, and temporary files. This occurs transparently, but "+" characters can be used, if desired, to get screen display of the various outputs. (An example of this is shown in Listing 5.)

Command lines

Command lines under MicroShell can be up to 84 characters long; and commands may have up to 17 arguments. Commands in a series are separated by semicolons. Obviously, very long commands can be subject to error when typed from the keyboard, so be careful. (Saving complicated commands as .SUB files helps!)

A neat answer to this problem is a built-in "edit" mode, invoked by typing

a single "!" from the prompt. This will fetch the last-used command line from the buffer, as well as place MicroShell in its edit mode. From this point, standard WordStar editing commands may be used to change the command. After it has been spiffed up, you can then re-execute it by hitting a return. Not only is this great for the editing, it is also of excellent utility just to be able to repeat the last command.

Shell files

All the things described thus far about MicroShell are certainly useful in themselves. But to my mind, the real power of the program lies in the shell files (.SUB files). Why? Simply because of their programming usefulness, which brings everything all together, adding synergism to boot.

As mentioned, shell files allow built-in SUBMIT-type processing, as well as the XSUB function. Shell files support the use of single or combined .COM files and intrinsic commands, as well as the extended shell commands. Used creatively, customized shell files allow some very useful functions to be implemented, for both the programming and the office environment. And they execute faster than do conventional SUBMIT files, since they are read into memory only once, initially.

Shell programming under MicroShell is supported by a number of flags, which act as toggles for certain functions or individual commands. For example, "–l" relogs disk(s); under Heath CP/M, it even allows you to change to disks of different density. Other examples are "–x", the exit to CP/M; "–v" (or "+v") turn off (or on) command echo; and "–s", shell-status report. Use of the –s command will yield a status report of the current shell conditions, as in Listing 2. (This example is for the default SH.COM, before any customization.)

Shell files support argument substitution, according to standard SUBMIT convention (as well as supporting the redirection and command-stacking features). Comment lines, if they are used, should begin with either ":" or ";". (See the examples in the various listings.)

The special extended shell commands begin with the "%" character (as we saw with "%clrstr"). As an example, "%print" is a string print command. Shell variables range from %A to %G (numeric) and %A$ to %G$ (string) and %0 to %18 (string argument substitution). Rather than go into the details of use at this point, some examples are more likely to give the general flavor.

Some examples

Listing 3, DEMOMENU.SUB, is a simple demonstration shell file which prints menu items 1-3 on the screen, and awaits your keyboard input. This illustrates a number of shell functions, and is a working example of how a menu might be used to foster friendliness. Obviously, it can be further customized. Also, you can use the string-print function to provide whatever level of help messages you wish.

```
-v
%erase
d
%print
%print
%print -n "Enter the filename to print: "
%getstr %B$ <$T
%2 = %B$
%print ^J
%print
%print
%print -n " Enter WS variation to use: "
%getstr %A$ <$T
%1 = %A$
$1
p $2^[x
;
; shut off cmd echo
; clear screen
; print spaces
; get keyboard input
; sub to pass fname
; prevent overprint
; you must have many versions, wslq wsdm, etc
; get keyboard again
; call up specified ws and print fname
;
;                          WSP.SUB
;                         08/08/83
;
;                       by Walt Jung
;
; This MicroShell sub file is called wsp.sub, and its function is
; to simply call a specified version of ws, with a keyboard entered
; filename.typ. Before doing this, it clears the screen, and prints
; the logged drive directory, then it prompts the user for a
; filename to be printed.
;
; After the filename has been entered, the desired ws version is
; prompted. It is called up, with the specified filename to
; print. No print options are requested, and the ESC question
; bypass is used. After printing, ws returns back to CP/M.
;
; Useful for a writer's environment, where multiple ws versions
; are used, with multiple printers.
```

Listing 4. WSP.SUB shows how MicroShell can be used to call up a program and print a specified file.

Listing 4 is an example of how MicroShell can be used to call up a program, in this case WordStar, and enter a file name to be printed. It allows printing the file via one of two (or more) configurations of WordStar, such as for draft- and letter-quality printers. Before prompting for your choices, it displays a sorted and sized directory on the screen. Choosing from this file directory, the operator simply enters a file name for printing. (More detailed comments are in the listing.)

Neither of these two particular examples is very technical. Indeed, their emphasis is really on ease of use, showing that anyone can use CP/M programs *without* getting overly technical.

Another example of a shell file is shown in Listing 5. And although this one is seemingly uncomplicated, it actually uses a great deal of the power of both MicroShell and the MicroTools. This one uses argument substitution to read and process a WordStar file ($1, FNAME.WS, for example). It produces a sorted and columnated word list ($2, FNAME.ANL).

Listing 5 is a good example of the use of pipes, using four of the MicroTools under MicroShell. DEFORM strips control and format commands from the file, places all words on separate lines, and passes the output to SORT, which sorts the words. UNIQ boils the list down to single occurrences. COL puts it in a four-column format, which gets sent to a file to be named FNAME.ANL.

So you end up with a four-column alphabetized list of all the words that appear in FNAME.WS. Quite a feat, when you consider the custom programming that would otherwise be needed to do this function. But it is a snap with the use of pipes and redirected I/O techniques.

This now leads us into the MicroTools themselves.

MicroTools

MicroTools, or "Unix-like utilities for CP/M" as the author calls them, provide many of the operational features of Unix. These include input/output re-

```
;          WORDANL.SUB
; Usage: wordanl fname.ws fname.anl
; Will read a WordStar file and analyze, using MicroTools
;
deform -x $1|sort|uniq|col>$2
```

Listing 5. A shell file which uses argument substitution, pipes, and redirection.

direction while operating from CP/M. And the general command syntax is in a Unix style. The listing of the utilities themselves is shown in part b) of Listing 1; a variety of functions are provided.

Although many of the MicroTools are intended to operate in concert with MicroShell, this is not a pre-requisite for their use. They are in fact highly useful in a stand-alone fashion under CP/M, as well as under MicroShell. Pipe operation in a CP/M environment is provided also.

Remember that it is a basic operating concept of the Unix environment to employ a number of "software tool" functions, in the form of Unix shell commands. MicroTools brings this concept to CP/M, with a series of individual program modules which perform specific functions, and are designed to talk to one another. (Again, the "C"-language/Unix special issue of *Byte*, August 1983, gives a lot of valuable background on these topics.)

The comments accompanying this article ("MicroTools—An Overview") describe the MicroTools in some detail. I think you will find they represent a quite flexible and powerful set of file-processing utilities, whether used alone or with MicroShell. Used with MicroShell, their utility becomes very broad—given all the options of the programs, and the ability to save unique option sets as shell files. They make a very useful set of tools, particularly for anyone with a lot of text-processing work.

Documentation

With both MicroShell and MicroTools, I found the documentation excellent in coverage and readability. Both manuals are nicely printed and easy to read. And they feature lots of examples

Some MicroShell Practicalities

As is noted in the body of the article, MicroShell overwrites that portion of memory occupied by the standard CP/M-80 CCP. It also takes up a portion of your transient program area (TPA), as well. This has both good and not-so-good ramifications.

On the good side is the fact that the need for the CP/M warm boot is a thing of the past, so long as you remain in MicroShell. With all of CP/M and MicroShell locked into memory, there is no need for the warm boot. There is not even any necessity for a system disk to remain in the A: drive! This is because MicroShell traps all program attempts to warm boot, and simply returns to the prompt, pronto.

This speeds up things in many instances, particularly with programs which would otherwise reload the CCP. And the execution of SUBMIT operations is greatly enhanced over the standard disk-based version—since the .SUB file is read into memory from disk just once. Then, the data can be processed amazingly fast. A shell file made up to call up DDT and patch a program, as but one example, can work in the blink of the eye, literally.

When operating under the shell, however, your useable program memory area is reduced by 8K. Logically, then, it behooves one to use all RAM possible (64K), to allow the maximum TPA. MicroShell can operate with less memory. But it just might not be able to run the very largest programs, or edit long files entirely in memory. In any case, there is no operator input required for loading MicroShell into memory, since it automatically sizes your memory when it loads, and protects itself as needed.

MicroShell has had extensive use on my H89, under two BIOS implementations (soft- and hard-sectored controllers). One is the C.D.R. Systems BIOS (versions 2.7C and 2.8C); and the other is Ray Livingston's CDRBIOS modifications to Heath 2.2.03. In addition, I have also used MicroShell with standard Heath CP/M 2.2.02 and 2.2.03 on H17 drives; as well as Magnolia CP/M 2.23+. I found no peculiarities from MicroShell with any of these systems.

Nor have I found any problems using it on the Z100 under CP/M-85—where it appeared to work just as well. It is worthy to note that the Z100 stands to gain the most from MicroShell—because of the Z100's extended TPA.

(On the H/Z89, you can already get a portion of some of MicroShell's features by using ZCPR, Richard Conn's public-domain CP/M CCP enhancement. It is for machines with the Z80 CPU—for those of you who are proficient in assembly language.)

of use to illustrate the various points. The MicroShell manual has an index, plus a number of appendices for more detailed information (such as custom patches to be installed with DDT, and bibliographic materials on Unix and "C").

A nice note is the sturdy and attractive binder furnished with the manuals. I absolutely abhor software houses which ship out a set of unbound looseleaf sheets with programs costing hundreds of dollars. I am pleased to note New Generation Systems has the priorities right on this point.

Summary

Using microcomputer software can at times be a trying experience, with serious bugs, program limitations, and poor vendor support. I am happy to state that MicroShell and the MicroTools showed none of these. In fact, they are a distinct pleasure to use. And they offer great utility as well as enhanced "user friendliness" over standard CP/M-80. I can recommend them highly.

To give some perspective on this, for quite some time prior to MicroShell I had been using the public-domain program ZCPR (Z80 Command Processor Replacement). Thus I was already accustomed to default .COM-file searches. MicroShell goes far beyond ZCPR in capability terms. For example, the shell-file customization features are simply an outstanding virtue, as is the batch processing.

For those using CP/M on hard disks, directory management can be a challenge. In addition to MicroShell and MicroTools, they should explore "MicroTree," a directory-management system available from New Generation Systems. MicroTree allows you to maintain hierarchical directories, to name directories, and to assign search paths. It brings these additional "Unix-like" features to CP/M and operates in a manner similar to the way MS-DOS 2.0 functions.

But perhaps most importantly to those with busy schedules, these programs are *supported*—in an active and positive fashion. Kudos go to Rick Rump and to Don Graft for two powerful, flexible, and highly useful packages. I recommend you check them out—you'll not only get more from your computer, you'll have loads more fun to boot!

Ordering Information

See Table 1.

The manual cautions the user that certain CP/M utilities (MOVCPM and SUBMIT) will not work under MicroShell. This is no real problem, since it would actually be unnecessary or illogical to use them. MicroShell has an inherent SUBMIT function; and MOVCPM requires a native CP/M environment.

To these two off-limits utilities, a Heath/Zenith user should add CONFIGUR—to avoid any system confusion in patching the CCP and BIOS.

Beyond these, all programs which use standard CP/M system calls should work, since that is what MicroShell looks to intercept. Exceptions may be found in some user-group programs. (Early versions of the sorted directory program SD, for example.) But from my experiences, incompatibility is rare.

I can see only one serious program pitfall with MicroShell. This is the potential size limit it might place on your programs. You might have problems when running very large programs. However, there may be an out, where even this tradeoff can be bypassed.

For the difficult situation where a big compilation requires the use of maximum memory, the manual shows examples of techniques where you can exit the shell for the duration of the compile; you can then re-enter it upon completion. This can be made automatic very simply.

Another example of a solution lies in the two related shell files of Listing 6. They can be used in conjunction with SUBMIT to load and run a large MBASIC program (greater than 22K in a 64K system). After the BASIC program has finished (with a SYSTEM command), the second .SUB file will automatically return to the shell.

```
; Call first four lines below 'MBAS.SUB'.
a:
%print a:sh>bak2ms.sub
submit mbas1 $1
-x
;
; Call these two lines 'MBAS1.SUB'
b:mbasic $1
submit bak2ms
; Usage: With SUBMIT on A:, MBASIC on B:, PROGRAM.BAS on A:
; command is of form 'MBAS PROGRAM'.
```

Listing 6. Using two shell files to exit MicroShell, run a large MBASIC program, and then return to MicroShell.

MicroTools—An Overview

The following is a brief overview of each MicroTools utility. Note that almost all of the MicroTools programs have a number of optional switches, which can modify operation beyond the basics described.

The name of the program and the form of its usage are given on the first line, then the description.

CAT cat [options] file_list [redirection]
CAT is a file-concatenation utility; it can send output either to the screen or to a single file (as PIP does). Wild cards are allowed. You can map non-printing characters (such as control characters, as well as WordStar high-bit flags) for optional printing.

COL col [options] input_file [output_file]
COL takes an input file and puts it into an output file (or the console). The output file appears in a multi-column format, in segments of 20 lines. A variety of output formatting options are provided, e.g., number of columns, page length, etc.

COM com [options] file1 file2 [output_file] [redirection]
COM compares two pre-sorted ASCII input files, and prints a three-column output. The first two columns contain the unique contents of the two input files, with the common words in the third column. Options are available for suppression of any column(s), and redirection is possible.

CRYPT crypt [–d] key input_file [output_file]
CRYPT does as the name implies—it encrypts and decrypts files. The operating mode is alterable with a switch. A user-specified key is used, on each of the two complementary steps.

CUT cut [options] input_file [output_file]
CUT will cut character or field columns from an input file, outputting them to a file (or the console). This is very useful for extracting single fields from multiple-field records, for example.

DEFORM deform [options] input_file [output_file]
DEFORM "deformats" formatting-command lines and in-line formatting characters which appear in text files. A →

Circle #116 on Reader Service Card

Circle #248 on Reader Service Card

Circle #123 on Reader Service Card

variety of options allow you to change the character or string used for a format. And you can use DEFORM to drop or map control characters. This can be very useful on WordStar files. (See the example of DEFORM's use in Listing 5.)

DIFF diff [options] file1 file2 [output file]

DIFF makes a line-by-line comparison of two files, and delivers an output which reports the differences, along with applicable portions of both files. This is very useful for comparing a revised file with a previous version. To see the changes under MicroShell, just send the DIFF program output to a file. (It is compatible with WordStar files, since it ignores the high bits set by WordStar.)

ECHO echo Argument to be echoed

ECHO simply echoes its own argument to the screen. This can be a string or an entire file. Within a batch file, ECHO can be used as an aid to documentation or tracking program flow. It functions much the same as the "%print" shell command under MicroShell (but does so from CP/M itself).

FIND find [options] input_file(s)

FIND will search for a pattern in a file, with various options. The user is prompted for the pattern, which can be a specific character string; or you can look for a more general pattern by mixing wild cards with specific characters. Input can be redirected. Options allow finding a pattern split across lines, line numbering, and others. Wild-card file names are allowed.

GET get –d or get –u

GET is a memory buffer which a MicroShell shell file can use as a scratchpad for a given drive or user area. You can change and restore the scratchpad's drive/user area.

GREP grep [options] pattern file(s)

(Use ~ before capital letters in pattern.)

grep [options] –p file(s)

(With the –p, GREP will prompt you for the pattern. In this form, the "~" is not required before capital letters.)

GREP, for "get regular expression," will find a specified pattern in a file, and display the lines containing it on the console. Wild-card file names are allowed. As noted above, you can either specify the pattern on the command line, or use a prompted mode.

A very flexible specification method allows you to find quoted or unquoted characters, and strings which have special characters for begin/end of line. Uppercase characters are preceded with a "~" in the command-line mode.

Options on GREP's operation allow line numbering, file naming and line counting, prompting, output to a file, "all-lines-except...", and "verbose" screen display (display of the entire file in which the pattern appears). GREP is one of the most powerful of the MicroTools, and is likely to be highly useful to those processing text.

LN ln file user_id

LN creates (or deletes) file directory links between CP/M USER areas for all file types. This allows a specified file to be made accessible in other USER areas, when placed in USER 0 for example. This saves disk storage, with the minor penalty of additional directory entries. Files "duplicated" in such a manner should be "erased" only with LN (not ERA).

MERGE merge [options] file1 file2 [output_file]

MERGE merges two pre-sorted input files, effectively extending the file-size capability of SORT to larger-than-memory size in the output file. Using the combined powers of SORT, SPL, and MERGE effectively, one can sort virtually any size input files (even larger than memory) by using sequential processing and pipes.

NEXT next list_file

(Input redirection is not allowed here.)

NEXT is a utility which allows looping within a shell file on a list of items. You can use it with FIND, first creating a list based on expanding a wild-card name. Then that output of FIND can be redirected to a list. Then a program called with a shell file can increment on that list. This is very handy, for example, to print a series of files.

P p [options] pipe_specification

When used in conjunction with CP/M's SUBMIT capability, P allows CP/M to implement the pipe function. Programs used with it must *themselves* support redirection. (P does not work under MicroShell—since it is not necessary.)

PASTE paste [options] input_files

PASTE concatenates files horizontally, as opposed to the more familiar sequential fashion. It is useful for merging columnar data, or combining separate fields. Options allow you to have variable-width columns and delimiters between the sections pasted together. And you can attach strings to act as prefixes and suffixes.

PR pr [options] input_file(s)

PR is perhaps the most flexible of all the MicroTools, and is a highly useful print formatter. A file can be printed either to the LST: device or to the console—with multiple copies, single or double spacing, page offset, TAB expansion, header specification, page numbering, and other options.

Wild-card file names are supported; e.g., pr*.txt>LST: will send all your .TXT files to the printer (with page numbering, default page-length, and the file names as page headings). This is a very useful utility—particularly under MicroShell, where custom options can be saved as shell files.

REC rec [options] input_file [output_file]

REC reformats multiple-field, single-line records into multiple-line records. The ":" is the default field delimiter that REC recognizes in the input file; but it can be changed. REC is useful for modifying the formatting of name lists and similar data.

→

SLEEP sleep n
(At 4 MHz, n is in seconds; and at 2 MHz, in 2-second units.)

SLEEP is a "do nothing for 'n' seconds" utility, useful for including pauses in shell file command sequences. Use n = 1 on 2-MHz machines for a two-second delay. (Sleep 30 = 1 minute.)

SORT sort [options] input_file [output_file]

SORT simply does what it says, sorting the lines of an input file and sending them to an output file (or to the console). Among its options, you can specify sorting a particular field in the input file, and you can reverse the sort order.

The entire file must fit in memory for SORT to operate, which will limit it on its own, as well as under MicroShell. (This limitation is functionally resolved with MERGE; see above.)

SPL spl [options] input_file

SPL is an inverse form of concatenation, splitting a file into two or more "chunks." Options include chunk by number of characters or lines, and chunk naming.

STR str [options] input_file [output_file]

STR is a string dump utility. It can be used to display printable ASCII strings within a binary file. Options include address labelling, and specifying starting addresses.

TEE tee file <pipe_out >pipe_in

TEE is a utility which will save the intermediate or "piping" results within a MicroShell pipe operation. This is useful for a partial repeat of a pipe operation, where a second pipe operation employs a different final part.

UNIQ uniq [options] input_file [output_file]

UNIQ filters out duplicate lines from a sorted input file. It sends the unique lines to an output file (or the console). It is useful in text analysis. (See example in Listing 5.)

WC wc [options] input_file(s)

WC counts the lines, words, and characters in an input file or files (wild cards supported). Options include character only, line only, word only, as well as options for formatting the output. Obviously, this utility can be of great value to a writer paid by the word!

U.S. Postal Service
STATEMENT OF OWNERSHIP, MANAGEMENT AND CIRCULATION
Required by 39 U.S.C. 3685)

1A. TITLE OF PUBLICATION	1B. PUBLICATION NO.	2. DATE OF FILING
Sextant	6 7 0 - 0 1 0	9/28/84
3. FREQUENCY OF ISSUE	3A. NO. OF ISSUES PUBLISHED ANNUALLY	3B. ANNUAL SUBSCRIPTION PRICE
bi-monthly	six	$14.97

4. COMPLETE MAILING ADDRESS OF KNOWN OFFICE OF PUBLICATION *(Street, City, County, State and ZIP+4 Code) (Not printer)*
716 E Street S.E., Washington, DC 20003

5. COMPLETE MAILING ADDRESS OF THE HEADQUARTERS OF GENERAL BUSINESS OFFICES OF THE PUBLISHER *(Not printer)*
716 E Street S.E., Washington, DC 20003

6. FULL NAMES AND COMPLETE MAILING ADDRESS OF PUBLISHER, EDITOR, AND MANAGING EDITOR *(This item MUST NOT be blank)*

PUBLISHER *(Name and Complete Mailing Address)*
Charles A. Floto, Sextant Publishing Co., 716 E Street S.E., Washington, DC 20003

EDITOR *(Name and Complete Mailing Address)*
Charles A. Floto, Sextant Publishing Co., 716 E Street S.E., Washington, DC 20003

MANAGING EDITOR *(Name and Complete Mailing Address)*
Jennie D. Allen, Sextant Publishing Co., 716 E Street S.E., Washington, DC 20003

7. OWNER *(If owned by a corporation, its name and address must be stated and also immediately thereunder the names and addresses of stockholders owning or holding 1 percent or more of total amount of stock. If not owned by a corporation, the names and addresses of the individual owners must be given. If owned by a partnership or other unincorporated firm, its name and address, as well as that of each individual must be given. If the publication is published by a nonprofit organization, its name and address must be stated.) (Item must be completed.)*

FULL NAME	COMPLETE MAILING ADDRESS
Sextant Publishing Company	716 E Street S.E., Washington, DC 20003
Charles A. Floto	716 E Street S.E., Washington, DC 20003

8. KNOWN BONDHOLDERS, MORTGAGEES, AND OTHER SECURITY HOLDERS OWNING OR HOLDING 1 PERCENT OR MORE OF TOTAL AMOUNT OF BONDS, MORTGAGES OR OTHER SECURITIES *(If there are none, so state)*

FULL NAME	COMPLETE MAILING ADDRESS
none	

9. FOR COMPLETION BY NONPROFIT ORGANIZATIONS AUTHORIZED TO MAIL AT SPECIAL RATES *(Section 423.12 DMM only)*
The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes *(Check one)*

☐ (1) HAS NOT CHANGED DURING PRECEDING 12 MONTHS ☐ (2) HAS CHANGED DURING PRECEDING 12 MONTHS *(If changed, publisher must submit explanation of change with this statement.)*

10. EXTENT AND NATURE OF CIRCULATION *(See instructions on reverse side)*	AVERAGE NO. COPIES EACH ISSUE DURING PRECEDING 12 MONTHS	ACTUAL NO. COPIES OF SINGLE ISSUE PUBLISHED NEAREST TO FILING DATE
A. TOTAL NO. COPIES *(Net Press Run)*	16,000	16,926
B. PAID AND/OR REQUESTED CIRCULATION 1. Sales through dealers and carriers, street vendors and counter sales	2,996	3,248
2. Mail Subscription *(Paid and/or requested)*	10,694	11,440
C. TOTAL PAID AND/OR REQUESTED CIRCULATION *(Sum of 10B1 and 10B2)*	13,690	14,688
D. FREE DISTRIBUTION BY MAIL, CARRIER OR OTHER MEANS SAMPLES, COMPLIMENTARY, AND OTHER FREE COPIES	593	533
E. TOTAL DISTRIBUTION *(Sum of C and D)*	14,283	15,221
F. COPIES NOT DISTRIBUTED 1. Office use, left over, unaccounted, spoiled after printing	1,236	1,705
2. Return from News Agents	481	0
G. TOTAL *(Sum of E, F1 and 2--should equal net press run shown in A)*	16,000	16,926

11. I certify that the statements made by me above are correct and complete — SIGNATURE AND TITLE OF EDITOR, PUBLISHER, BUSINESS MANAGER, OR OWNER: Charles A. Floto Publisher

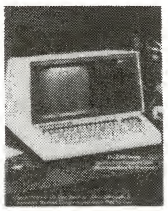
PS Form 3526, July 1984 *(See instruction on reverse)*

Sextant BACK ISSUES are loaded with articles about your Heath/Zenith computer. Pick up the issues you've missed—and get *all* the information about how your computer system can work even better for you!

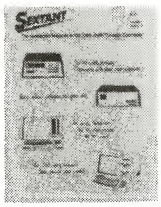
Here are some of the highlights of each issue:

☐ **Spring '82 (#1)**
• A Parallel Interface for the '89 • Disk Programming *Without* HDOS • Print Spoolers: A Background and Reviews • A Professional Writer Looks at Budget Word Processing • A Candid Look at Tiny Pascal

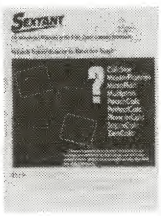
☐ **Summer '82 (#2)**
• Crucial Secrets of HDOS Let You Write Device-Independent Programs • Preview of the Z100 • Put Some Structure in Your BASIC Programming • Renumbering Benton Harbor BASIC



☐ **Fall '82 (#3)**
• The World's Fastest Sort? • Give Your '89 a Boot—Automatically! • Double-Precision Trig Functions • Graphics on Your H8



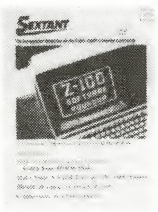
☐ **Winter '83 (#4)**
• Interrupt Programming Under HDOS • Eight Spreadsheets Compared • Venturing Beyond the Heath/Zenith Disk Controller • A Survey of Disk Controller Alternatives



☐ **Spring '83 (#5)**
• CP/M: The First 13 System Calls • Understanding HDOS • Improve Your H8's Reliability: Install the Trionyx Motherboard • Improve Your H8's Reliability: Add a Fan

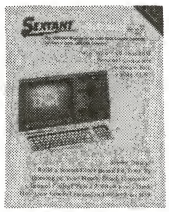
☐ **Summer '83 (#6)**
• Pascal Elucidated • Assembly Language Inside a Spooler • Squeeze More Disk Space Out of HDOS • Taking a Look at Magnolia's Invisible Disk • Passwords and Record-Keeping

☐ **Fall '83 (#7)**
• Z100 Software Roundup—over 80 products for the Z100 • Make Your '8 Think Like an '89—Only Faster • dBASE II Comes to Heath/Zenith • Index Your HDOS Disks



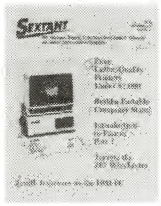
☐ **Winter '84 (#8)**
• Games Roundup—nearly 50 games for Heath/Zenith computers • Getting Into Z-DOS Assembly Language—Quietly • Kids and Computers • Tracking Home Finances With the Home Finance System

☐ **March-April '84 (#9)**
• Why Won't All That IBM-PC Software Run on My Z100? • Screen Dump! • More on HDOS STAND-ALONE • A Z100 on Every Desk: Clarkson College • Build a Sound/Clock Board for Your '89



☐ **May-June '84 (#10)**
• Educational Software Roundup • Windows Opens the Door to Compatibility • Color Graphics for Your '8 or '89 • Put an Electronic Disk into Your H8 • Graphics Algorithm Optimized for 8080 or Z80

☐ **July-August '84 (#11)**
• Review of the IBM-compatible Z150 • Introduction to Pascal: Part 1 • Four Letter-Quality Printers Under $1000 • The Z67 Winchester: Use It With HDOS as Well as CP/M • Build a Portable Computer Stand



☐ **September-October '84 (#12)**
• Roundup of Z-DOS Communications Software • Introduction to Pascal: Part 2 • Communing By Computer—the HUG Bulletin Board • Forth Goes Heath/Zenith • Index to *Sextant* #1–#12

☐ **November-December '84 (#13)**
• Piggyback a Parallel Port on Your '89 • Introduction to Pascal: Part 3 • Making Z-BASIC Friendlier • Megabyte Memory for Your H8 • Moving Your Old CP/M Files to Your New Computer

Order your *Sextant* back issues today!
Return this coupon to:
Sextant
Dept. 14
716 E Street S.E.
Washington, DC 20003

Send me the issues I've checked:

☐ Spring '82
☐ Summer '82
☐ Fall '82
☐ Winter '83
☐ Spring '83
☐ Summer '83
☐ Fall '83
☐ Winter '84
☐ March-April '84
☐ May-June '84
☐ July-August '84
☐ September-October '84
☐ November-December '84

$3.50/issue × _____ issues = $ _____.
Please enclose payment with order. (Please include $1.50 additional for each issue to be mailed overseas.)
☐ Payment enclosed
☐ Charge my: ☐ Visa ☐ MasterCard

Card # _____ Expires ____

Issues will be mailed first class upon receipt of payment.

Name _____

Address _____

_____

Clip this coupon and send it today to:
Sextant, Dept. 14, 716 E Street S.E., Washington, DC 20003

Circle #103 on Reader Service Card

Circle #192 on Reader Service Card

Circle #135 on Reader Service Card

Circle #169 on Reader Service Card

Circle #151 on Reader Service Card

Supplier Notes

"Near Letter Quality" Dot-Matrix Printer from M.P.I.

Micro Peripherals, Inc., has begun shipping their new "X"printer, a dot-matrix printer that can produce results "near letter quality." The "X"printer works with all Heath/Zenith systems.

The "X"printer is a bidirectional, 80-column printer that provides draft-quality printing at 180 characters per second (cps), correspondence mode at 120 cps, and "executive mode" at 50 cps.

The print pitch of the "X"printer can be adjusted to 10, 12, 14.4, 15, or 17 characters per inch, and various character fonts may be downloaded through the printer's firmware. The "X"printer also prints international characters, superscripts, subscripts, and images. It weighs 16 pounds. And for a limited time a travel cover for portability is available at no charge. The unit comes standard with friction and tractor feeds and AutoSheet load. AutoSheet load will automatically set a single sheet in the correct position for printing.

Optional features of the "X"printer include Repeat mode and SideWinder mode, plus a 64K buffer (required for the use of Repeat and SideWinder). The buffer allows spooling of roughly 34 pages of text in the printer. Repeat mode, when activated by the operator, can reprint a document, without computer tie-up, up to 99 times. SideWinder mode permits the user to rotate text or other material 90 degrees. This sideways rotation gives almost unlimited printing width without changing the host computer software.

For further information, contact M.P.I., 4426 South Century Drive, Salt Lake City, UT 84123, 801/263–3081.

World Map Projection Software

High-resolution maps may be drawn of the world coastlines or of the Caribbean area specifically with software entitled "Map Projections of the World."

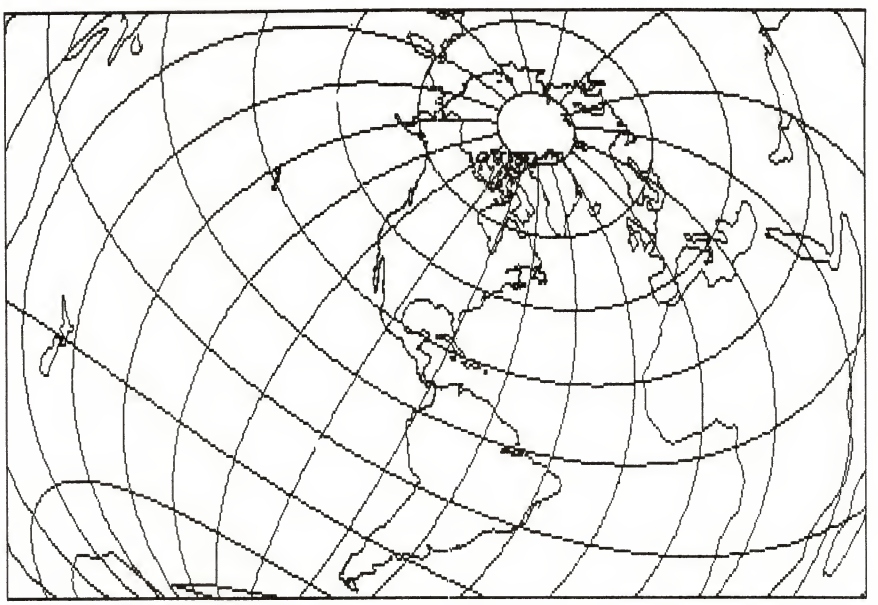
Seven different projections (polyconic, Mercator, and five azimuthal projections) may be selected. The user can also choose any origin on earth, its location on the screen, the scale and rotation angle, and the intervals for the grid lines (meridians and parallels).

On some projections, one may view up to ten hurricanes in animation or track them across the screen. Two historical storms are built in, and more may be added by the user.

The documentation provides the necessary input data to reproduce 16 sample maps as the user learns about the various projections.

The minimum requirements are an H/Z110-120 or H/Z150-160; a version of MS-DOS (Z-DOS or PC-DOS); and color graphics capability. A color monitor is recommended.

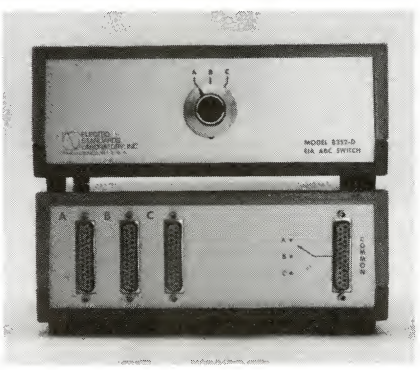
The program is available at many Heathkit Electronic Centers, or from the author. To order direct, send check or money order for $49.95 to Dr. M.E. Pittman, 57 Emile Ave., Kenner, LA 70065.



Three-Way Port Sharing Switch

Electro Standards Laboratory, Inc., of Providence, Rhode Island, has introduced its Model 8351-D and Model 8352-D EIA RS-232 A,B,C Switches. They are designed to improve computer-room efficiency by permitting one computer port to be shared among three peripherals.

This product allows three terminals to use one I/O port, three CPUs to share one printer, or three terminals to share one modem. The Model 8351-D switches twelve of the most required EIA RS-232 interface signals and therefore is sufficient for most port-sharing applications. The Model 8352-D switches all signals at the EIA RS-232 interface and therefore will satisfy any three-to-one sharing application. The devices are housed in a desktop case. All connections are made via four female 25-pin EIA connectors on the rear panel. The connectors are labeled A, B, C, and Common. A switch on the front panel will route signals from the Common connector to either the A, the B, or the C connector.



Price of the Model 8351-D is $160; price of the Model 8352-D is $180. Send inquiries to Electro Standards Laboratory, Inc., P.O. Box 9144, Providence, RI 02940, 401/943–1164. Cable address: ELECTROSTD Providence, RI, U.S.A.

Classified Ads

Hardware

ASSEMBLED H25 PRINTER, $500.00 plus shipping. Robert Crowell, 7040 Highway 99 South, Redding, CA 96001, 916/246–1633.

5 MEG HARD DISK SUBSYSTEM for H89/90. Kits from $800. Digital Search, 103 South Main Street, Greer, SC 29651, 803/877–9444.

Z100 ACCELERATOR! Drive your 8088 CPU at a true 8 MHz, increase program and graphics execution by 50%! Works with all 150ns RAM and most 200ns chips. Our easily installed modification involves changing one crystal on the motherboard. $24.95 postpaid. Human Interfacing, 4309 S.W. Cullen Blvd., Portland, OR 97221, 503/245–9543.

Software

ENGINEERING SOFTWARE for H/Z 89/90/100/150. Write for free catalog. BV Engineering, 2200 Business Way #207, Riverside, CA 92501, 714/781–0252.

FLOPPY DISKS and other supplies at Discount Prices. Call 212/652–2053, 7 to 11 p.m. Eastern time. MATCOMM, POB 604, Bronx, New York 10469.

GHASTLY MANOR—game disk $5.00 (DS/DD/soft—specify CP/M, MS-DOS, Z-DOS). Self-addressed stamped envelope for catalog describing several games, I-CHING (the original computer oracle ... in a fresh translation ...), BASFORM (programmer's utility), more.... Dragon Tree, 1310 College Ave., Suite 1151, Boulder, CO 80302.

To get in touch with your fellow Heath/Zenith users, place your short notice in *Sextant*'s classified section. The rate is only 75¢ per word with a minimum of 15 words. Send your typewritten ad and payment to: Sextant, Classified Ad Department, 716 E Street S.E., Washington, DC 20003. (Please include your name and telephone number for our records.) We need to receive ads for the March-April issue by January 11, and for the May-June issue by March 8.

Coming up in *Sextant*:

- how to build an automatic screen-brightness controller for your '89
- how to create a Greek (or other alternate) character set for your Z100
- a review of Galahad, Clarkson College's word-processing program for the Z100
- a report on the Air Force Small Computer Conference, which Zenith computers dominated
- a review of Commsoft's ROOTS genealogy program for the H89
- a review of Redding Associates' Graftalk business graphics program for the Z100 and Z150
- a look at how some people are using Heath/Zenith computers in research and development

Luminaries

James Bingham is an electronics technician working on microprocessor-controlled machinery. He told us about his MPI printer in the Winter 1984 issue of *Sextant*. So now he's filling out the story with information on his interface to the printer.

Roy Coleman teaches physics at Morgan Park High School in Chicago. He's been playing with computers for 20 years, and with Heathkits for 12. We're wondering if he's had as much success getting his students to process his commands as he's had with HDOS.

Raymond Dotson spends 5–6 hours per day at the keyboard "cranking out the programs." That makes for a lot of programs—especially when they're only five lines long. MARQUEE is the fifth BASIC program he's had in *Sextant*.

M. H. Endres is a Systems Engineer for Kentron International working atop Mt. Haleakala on the island of Maui in Hawaii. He's been active in personal computers since 1973, when he built his first one using Intel's 8008 chip. He built his '100, too. Lucky for him, as a programmer, that his only real problem with the computer was overcome with a short Z-BASIC program.

David R. Felstul was an intern at *Sextant* last summer. During that time he reviewed everything he could get his hands on. Dave goes to St. Olaf College, in Northfield, Minnesota, which has over 50 H89s and several H100s. Sounds like pretty good training to become a Heathkit hacker.

Walter G. Jung is a consulting author living in Maryland. He's written several books on integrated circuits. Turning to software, he's chosen an equally mysterious subject to elucidate—the goings-on in CP/M's CCP.

Beaufort Lancaster is a Senior Project Engineer at the Frito-Lay headquarters in Dallas. He has programming experience in several languages on many brands of mainframes and minis; and his latest hobby interest is learning C. Needless to say, we applaud his choice of word processor.

Wayne Rash, Jr. is a consultant with American Management Systems in Arlington, Virginia. He's involved with helping people use microcomputers. So he's used to reviewing computer products. This is his fifth time doing it for us.

Donald E. Risher is a meteorologist with the National Weather Service in Charleston, West Virginia. His interest in computers has progressed through digital electronics, BASIC, and assembler, to Pascal. He has since progressed beyond the subject of his article, having just "discovered" Turbo Pascal.

Arthur L. Thomas is Arthur Young Distinguished Professor at The School of Business, The University of Kansas at Lawrence, and an inveterate end user. Managing 600+ students and a 23-person staff, he can't afford to waste time debugging programs.

Circle #192 on Reader Service Card

Circle #230 on Reader Service Card

Circle #221 on Reader Service Card

Two Debugging Tricks for MBASIC

RENUM and BASCOM will quickly identify trivial programming errors. After you've run them, you can look for the real *bugs.*

Arthur L. Thomas

Like most people, when programming in Microsoft BASIC (MBASIC) I make lots of errors that eventually must be corrected. Most of these are trivial: mistakes in typing, forgetting to close parentheses, failing to change GOTO and GOSUB references to renumbered lines, and the like.

If the program is short, such errors are easy to detect when running it under the MBASIC interpreter. But in a lengthy program, they can remain undetected for a long time, especially if they:

1. are in sections of code that the program rarely uses; or
2. involve line references from previously debugged sections to other sections whose subsequent debugging includes line-number changes.

Some languages, and some dialects of BASIC, inform you of most such errors as soon as you try to run your program. But interpreted MBASIC tells you about only one error at a time, when it tries to execute the line in which the error occurs.

Yet it is easy to sweep through an entire, lengthy MBASIC program, without running it, and detect most such trivial errors at the start of debugging. After correcting them, they won't distract you when you search for serious errors that result from things like faulty algorithms.

MBASIC provides you with two features that can be used to accomplish this preliminary debugging. The less powerful of these tools is the RENUM command; it will find references to nonexistent lines. The more powerful technique uses the compiler, BASCOM, at a much earlier stage in program development than usual.

An example

Both tools are best explained by working on an example. The program in Listing 1, FRIENDLY.BAS, parodies a user-friendly program. Its first section does a little housekeeping and then calls its second section, which elicits a name from the user and tests that name for length. If the name is too long, the third section records a bad reply, then cautions the user. Or, if there have been too many bad replies, it gives up and calls a mildly abusive fourth section.

FRIENDLY.BAS is short enough that any trivial errors are easily found. But it might be the introduction to a program hundreds of lines long.

Using RENUM

Let's begin by modifying FRIENDLY.BAS so that it makes two references to nonexistent lines. Change its lines 1070 and 2030 to:

```
1070 IF BADREPLY = 3 GOTO 1120
2030 GOSUB 4000
```

Load the modified program and LLIST your code to the printer for a record. If

```
 10 ' FRIENDLY.BAS
20 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
30 '
40 ' Initialize program:
50 BLANK$ = CHR$(27) + "E":  ' Erase screen.
60 BUZZ$ = CHR$(7):  ' Sound buzzer.
70 BADREPLY = 0:  ' Counter of bad replies.
80 GOSUB 1000:  ' Elicit user's name.
90 END:  ' A real program would instead call another subroutine.
980 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
990 '
1000 ' Elicit user's name:
1010 PRINT BLANK$
1020 PRINT:  PRINT
1030 PRINT "Hi.  What's your name?"
1040 INPUT "(Please use no more than 12 characters)"; REPLY$
1050 IF LEN(REPLY$) < 13 GOTO 1090:  ' Name is okay.
1060 GOSUB 2000:  ' Name is too long.
1070 IF BADREPLY = 3 GOTO 1110:  ' Program has given up.
1080 GOTO 1040:  ' Try again to elicit appropriate name.
1090 USER$ = REPLY$
1100 PRINT:  PRINT "Thank you. "; USER$; " is a neat name!"
1110 PRINT:  PRINT:  RETURN
1980 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1990 '
2000 ' Name is too long:
2010 BADREPLY = BADREPLY + 1:  ' Count number of bad replies.
2020 IF BADREPLY < 3 GOTO 2050:  ' Give user another chance.
2030 GOSUB 3000:  ' Too many bad replies;  stop seeking name.
2040 RETURN
2050 EXCESS = LEN(REPLY$) - 12:  ' Calculate number of
2060                             '    excess characters.
2070 PRINT:  PRINT BUZZ$
2080 PRINT "Sorry, but that's"; EXCESS; "too long."
2090 RETURN
2980 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2990 '
3000 ' Too many bad replies;  stop seeking name:
3010 PRINT
3020 PRINT "Obviously you are some sort of wiseguy or something."
3030 PRINT "     Okay, wiseguy, I will call you 'Snookums'."
3040 USER$ = "Snookums"
3050 RETURN
```

Listing 1. FRIENDLY.BAS, a parody of a user-friendly program. By making changes to FRIENDLY.BAS and running RENUM or BASCOM, we can see how they identify typographic errors and references to nonexistent lines.

Circle #219 on Reader Service Card

Circle #210 on Reader Service Card

Circle #201 on Reader Service Card

you then command RENUM, MBASIC will immediately tell you that there is an unidentified line 1120 in 1070, and an unidentified line 4000 in 2030. As it's in memory, the program is now a mess since it's partially renumbered and partially not. Get rid of it with NEW or SYSTEM. You can then use your listing to correct the line references.

To be sure, this works only with references to nonexistent lines. If line 1070 mistakenly read GOTO 1010, you wouldn't find the error until you ran the program.

Using BASCOM

The BASCOM compiler will also detect references to nonexistent lines. Once again, FRIENDLY.BAS should incorporate the errors used above. If it is on your B: disk and BASCOM on A:, enter:

```
BASCOM=B:FRIENDLY/S
```

After a few moments, BASCOM will tell you that line 1120 is undefined, line 4000 is undefined, and that these are severe errors. You can then delete the resulting file, B:FRIENDLY.REL, and use your text editor's search function to find references to lines 1120 and 4000.

This isn't quite as efficient as the RENUM way, but BASCOM can detect a wider variety of mistakes. To illustrate this, let's modify FRIENDLY.BAS to contain the following typographical errors:
Line 1050: Omit the) that terminates LEN(REPLY$);
Line 3000: Omit the initial ' remark indicator;
Line 3010: Change PRINT to PTINT.

BASCOM will tell you that there is both a type mismatch and a syntax error in line 1050, and syntax errors in lines 3000 and 3010. Similarly, if you typed a comma instead of a < in line 1050, or left out the $ in REPLY$, BASCOM would tell you of your sins.

As before, this technique won't detect *all* trivial errors. For example, suppose that you accidentally omitted the first " in line 3020. The interpreter will perceive the erroneous line to ask for the values of a series of single-precision variables, and will display seven zeros.

A compiler only tells you when an interpreter can't function, not when it's doing something you didn't want. So, BASCOM won't detect anything wrong here. Nor would it catch our earlier erroneous reference to line 1010.

Summary

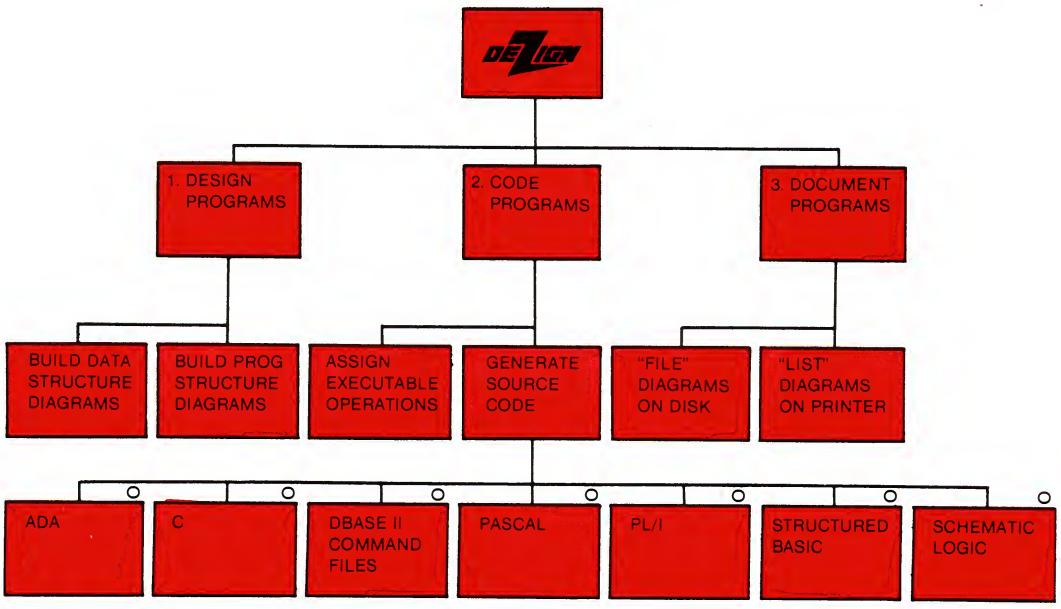
Despite such limitations, both RENUM and BASCOM are powerful debugging tools. I use RENUM first, to find references to nonexistent lines. One might expect to compile MBASIC programs only after thoroughly debugging them. Yet use of BASCOM immediately after RENUM can protect your serious program testing from repeated interruptions by trivial errors. I haven't tried it yet, but the same techniques should work with Z-BASIC and *its* compiler.

Use RENUM, then BASCOM, as soon as you've typed your program in. Correct the errors that they detect. Then, knowing that your program will run, settle down to the more important job of making sure that it runs just the way that you *intended*.

EVERY PROGRAMMER/SYSTEMS ANALYST NEEDS . . .

DEZIGN

. . . THE STRUCTURED PROGRAMMING TOOL FOR THE 80s!



Design your programs right on the screen, using modern structured programming techniques based on the Jackson Program Design Method (JSP)! **DEZIGN** is truly the programmer/analyst's friend – more than just another flowcharting tool!

DEZIGN – an integrated tool for designing programs, generating source code in the language of your choice, and documenting your programs for future reference! Even non-programmers will enjoy using **DEZIGN** to build family trees and organization charts!

Available, as follows, from Zeducomp and Heath/Zenith dealers:

			Column A	Column B
☐ DEZIGN	Z-89	CP/M (Hard sector)	$ 70.00	$140.00
☐ DEZIGN	Z-90	CP/M (Soft sector)	$ 70.00	$140.00
☐ DEZIGN	Z-100	CP/M-85	$ 70.00	$140.00
☐ DEZIGN-86	Z-100	CP/M-86	$100.00	$200.00
☐ DEZIGN-16	Z-100	Z-DOS	$100.00	$200.00

Column A Prices include one Source Code Generator of your choice; column B prices include all Source Code Generators shown above.

ALSO AVAILABLE FOR THE IBM PC AND ZENITH Z150!

ZEDUCOMP

P.O. BOX 68 ● STIRLING, NJ 07980 ● (201) 755-2262

JSP is described in "Principles of Program Design", by Michael Jackson, Academic Press.

CP/M is a trademark of Digital Research, Inc. dBASE II is a trademark of Ashton-Tate, Inc. Z-DOS is a trademark of Zenith Data Systems.

Circle #236 on Reader Service Card

Scuttlebutt

Rumors from Zenith Data Systems: We've heard that ZDS is currently working on an IBM **PC AT clone**. (Are we surprised?)

Rumor also has it that Zenith will be marketing (we use the term loosely) a **lap-size** computer within the next few months. The computers will be manufactured in **Japan**. Does that mean the quality will go in in Japan, and the name will go on back here?

Whereas we correctly label rumors, *InfoWorld* prints them (**garbled interpretations**, anyway) as fact. Their October 29 issue had a paragraph on the Air Force/Navy contract awarded to ZDS for **Tempest** Z150s. **Yes**, the contract has been awarded. **However . . .**

InfoWorld said $100 million worth of Z150s had been **sold**. Actually, that amount may be spent on them over **the next two years**. As of this printing, no money has changed hands.

A little bit of arithmetic yielded for *InfoWorld* the sum of 10,500 computers at about **$10,000 each**. The correct figures are $2,899 (with floppies) or $4,699 (with hard disk). The rest of the contract is for **peripherals** such as printers and modems.

They explained "**Tempest**" as "the code word for a microcomputer modified so that it doesn't emit radio waves." The term Tempest can apply to **any electronic apparatus**, not just microcomputers. And the radio waves they want to prevent are those that could be read for **information**. White noise they don't mind.

Before Zenith presented a new design for the Tempest Z150, Tempest machines were **encased in a lead shielding**, *InfoWorld* reported. Actually, shielding was previously done with copper—and usually inside the machine. Zenith's new design secures emissions **electronically**.

Last year, according to *InfoWorld*, Zenith "sold $30 million worth of their Z100 computers to the military's **school system**, a neat trick considering the Z100 is a **CP/M computer** for which there isn't much in the way of educational software." Did they get **anything *right*** in that sentence?

All that, in **only half a dozen sentences**. Well, they printed a correction in the following issue, titled "Zenith defenders fight back (and jump down editor's throat)." Zenith users do know how to carry on.

It's **conference season** again. We were at the **Air Force Small Computer Conference** in Montgomery, Alabama, in October. It was nice to be at a show where the crowds around the Zenith booth were of the size you usually only see around IBM displays. We'll have a full report on that in our March-April issue.

We also went to the Capital Heath Users' Group Conference and the West Coast Regional HUG Conference. You may be wondering how, with all these shows to go to, we find time to put out a magazine. Well, **we're wondering too**.

We forgot to tell you. . . . If you're wondering what happened to **Ziggy**, Zenith's "public relations man"—he was "let go" last spring. Apparently Zenith found the contract too limiting. And, according to Bob Winter, director of marketing at ZDS, "we wanted more of a hard hitting, direct approach in our ads to combat the IBM PC." We're waiting. . . .

Get out that diploma. . . . We've heard that Zenith is opening up their **discounts** to **alumnae** of the colleges and universities under contract for Z100s.

In his new book (reviewed on page 31 of this issue), **Hugh Kenner** included a chapter (a full page in length) on **publications and support** for Z100 users. In it he explains the reason for *Sextant*'s name—"because a nautical sextant locates heavenly bodies with respect to the zenith."

The advantages of being a kit builder. . . . Here at *Sextant*, we were unable to hook up our new **Z150** to a printer last spring. The connection requires a null modem cable, which we looked high and low for and were finally forced to obtain from outside the Heath/Zenith community. Well, Mike Cogswell of CHUG has provided the answer. The needed cable *is* available from Heath—and it **comes with the H150** as part of the test setup.

Check Out the Entire IBM/PC Library With Your Zenith Z100 System

That's right. The new Gemini Emulator Board now makes the Zenith Z100 family of computers compatible with the entire IBM/PC library of programs — all 5,000 of them.

And what's more, there is no loss of computing power and no loss of time. Just plug in the Gemini Emulator Board and go to work.

It instantly selects the proper mode for the program you choose to run whether it is one of your existing programs or any program from the IBM/PC library.

Gemini Emulator Boards are products of :

D.E.L.
PROFESSIONAL SYSTEMS INC.
12-11151 Horseshoe Way
Richmond, B.C. V7A 4S5

Get full details on the Gemini Emulator Board at your local Heathkit Electronics Center.

Circle #178 on Reader Service Card